GENERALIZED EDGE COVERING
AND
RELATED PROBLEMS

By

SUNEETA AGARWAL

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

DECEMBER, 1979

GENERALIZED EDGE COVERING AND RELATED PROBLEMS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

By

SUNEETA AGARWAL

to the
DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
DECEMBER, 1979

MATH-1978-D-AGA-GEN.

CERTIFICATE

This is to certify that the work embodied in the thesis 'GENERALIZED EDGE COVERING AND RELATED PROBLEMS' by Suneeta Agarwal has been carried out under our supervision and has not been submitted elsewhere for a degree or diploma.

A K. Mittal
Assistant Professor

Prabha Sharma
Assistant Professor

Indian Institute of Technology
Kanpur-208016

December - 1979

ACKNOWLEDGEMENTS

I am deeply indebted and would like to express my heartfelt gratitude to my supervisors, Dr. A.K. Mittal and Dr (Mrs.) P. Sharma who have constantly encouraged and guided me throughout the preparation of thesis.

Words can not express my sense of indebtness to my grandfather who was perennial source of inspiration and encouragement.

I am also thankful to Mr. S.K. Tewari and Mr. G.L. Misra for typing the manuscript.

December - 1979 *Suneeta Agarwal*

CONTENTS

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION

In recent years considerable attention has been paid to the specially structured linear integer programming problems defined on graphs. "Good" algorithms as defined by Edmonds [15] have been reported for some of these problems. In this thesis we will concentrate on a generalized version of one such problem. In this chapter we will give a brief introduction of the problems of this class and later, a chapterwise brief summary of the remaining chapters of this thesis.

Let $G \equiv (N,E)$ denote a finite, undirected and loopless graph where N and E are node and edge sets of G respectively, and $|N| = n$, $|E| = m$. C is an n x 1 cost vector associated with the node set N of G. The problem of covering of edges by nodes, is to select a subset T of N, which covers all the edges of G and which also minimizes the cost, over all such covers of E. A node is said to cover all the edges of G, which are incident upon it. Above problem is referred to as "Weighted Edge Covering Problem (CP)". For nomenclature of this problem and other problems of this class we will follow Balinski [7]. CP is formulated as the following integer program by representing a cover T by a binary vector U, such that $u_p = 1$ if and only if $p \in T$:

CP :

$$\begin{aligned} &\text{minimize} \quad CU \\ &\text{such that } AU \geq e \\ &\qquad u_p = 0,1 \quad \forall\, p \in N \end{aligned}$$

where,

A is an edge-node incidence matrix of G and e is a summation vector.

Let CLP denote the linear relaxation (i.e. the relaxation of integrality constraints) of CP. Dual of CLP is

MLP :

$$\begin{aligned} &\text{maximize} \quad e\,Y \\ &\text{such that } Y\,A \leq C \\ &\qquad Y \geq 0 \end{aligned}$$

which is a linear relaxation of the following C-matching problem (MP) :

MP :

$$\begin{aligned} &\text{maximize} \quad e\,Y \\ &\text{such that } Y\,A \leq C \\ &\qquad y_{pq} = 0,1 \quad \forall\, (p,q) \in E. \end{aligned}$$

Another problem which is equivalent to CP, and which is also of great interest is the "Vertex Packing Problem (VP)". Vertex packing problem is to find a subset I of N with the highest cost such that no edge of G is incident upon two nodes of I. It is formulated as the following 0-1 programming problem :

VP :

$$\text{maximize} \sum_{p=1}^{n} c_p u_p$$

$$\text{such that } u_p + u_q \leq 1 \quad \forall (p,q) \in E$$

$$u_p = 0,1 \ \forall p \in N$$

VP is equivalent to CP in the sense that solution of one can be obtained from the other by a simple linear transformation $U = e-U$. Let VLP denote the linear relaxation of VP.

Dual of VLP is

$$\text{minimize} \sum_{(p,q)\in E} x_{pq}$$

$$\text{such that } \sum_{q\in N} x_{pq} \geq c_p \qquad \forall p \in N$$

$$x_{pq} \geq 0 \qquad \forall (p,q) \in E$$

For the special case when $c_p = 1 \ \forall p \in N$ above problem together with the integrality conditions on x_{pq}'s is known as the "Covering of Nodes by Edges" [7]. Edmonds and Johnson [16] proposed the following general problem :

$$\text{minimize} \sum_{(p,q)\in E} c_{pq} x_{pq}$$

$$\text{such that } b_1 \leq AX \leq b_2$$

$$x_{pq} = 0,1 \quad \forall (p,q) \in E$$

which obtains a maximum matching or minimum covering as a special case.

White [58] discussed the problem of maximum matching and minimum weighted covering of fixed cardinality.

1.2 SURVEY OF RELATED PROBLEMS

Considerable work has been done on these closely related problems. Berge [8] has proved the connection between minimum cardinality cover (when $c_p = 1$, $\forall\, p \in N$ in CP) and a maximum cardinality matching (when $c_p = 1$, in MP $\forall\, p \in N$). He introduced the notion of alternating and augmenting paths and proved that a matching is of maximum cardinality if and only if no augmenting path exists. Similarly for minimum covering problems Norman and Rabin [45] defined alternating and reducing paths. Following are some of their main results :

Theorem 1.2.1 : Every irreducible cover is a minimum cover.

Theorem 1.2.2 : If M_1 and M_2 are two covers then M_2 can be obtained by applying a finite number of linear transformations on M_1.

These notions are generalized for weighted covering and matching problems by Witzgall and Zahn [59] and Balinski [7]. Edmonds [14] has developed algorithms to search these augmenting and reducible paths. Witzgall and Zahn [59] and Balinski [7] presented simple alternatives to this search procedure, but these simplifications appear to be effective only for the problems when C is a constant multiple of the summation vector.

Vertex packing problem has recently received considerable attention. Properties of the convex hull of feasible solutions of VP have been investigated by Chavatal [12], Padberg [47],

Nemhauser and Trotter [43] and Trotter [55]. Algorithmic aspects of VP have also been studied by Balinski [7], Trotter [55], Balas and Samuelson [5].

It has also been shown that the vertex packing problem and hence the weighted edge covering problem are "Complete" in the sense of Karp [30].

Nemhauser and Trotter [43] introduced the new idea of solving VP by a decomposition method which is not possible for arbitrary integer programming problems. They have also developed some useful optimality conditions which can be summarized as follows :

Theorem 1.2.3 [43]: Let V be a vertex packing of G. Let $\bar{V} = N/V$, then V is an optimal packing in G if and only if every maximal packing $I \subseteq \bar{V}$, $V(I)$ is an optimum packing in the bipartite subgraph $\hat{G}$ of G, induced by $I \cup V(I)$, where $V(I) = V \cap N(I)$ and $N(I) = \{p \in N/I : (p,q) \in L \text{ for some } q \in I\}$.

The above optimality condition then gives the following 'local' sufficient condition for optimality :

Theorem 1.2.4 [43] : If V is an optimal packing in $\hat{G}$, induced by $V \cup N(V)$, then $V \subset V^*$, where V^* is an optimal packing in G.

By using the special structure of its constraint set, they have proved the following result which is useful for decomposing the problem into two parts.

Theorem 1.2.5 [43] : Suppose X^* is an optimal $(0, \frac{1}{2}, 1)$ valued solution of VLP and $\bar{V} = \{p \mid x_p^* = 1\}$. There exists an optimum packing in G that contains $\bar{V}$.

The algorithmic effectiveness of the above result is enhanced by the fact that VLP is equivalent to VP on a bipartite graph G' and therefore can be solved by a good algorithm. This is a result by Edmonds and Pulleyblank, details of which is available in [43]. Following is the restatement of this result which represents a natural correspondence between feasible solutions to VP on G' and feasible $(0, \frac{1}{2}, 1)$ valued solutions to VLP on G, that preserves objective values appropriately.

Theorem 1.2.6 [43] : (i) If (U, U') is a feasible solution to VP on G', then $\bar{U} = \frac{1}{2}(U+U')$ is a feasible $(0, \frac{1}{2}, 1)$ valued solution to VLP on G.

(ii) If $\bar{X}$ be a feasible $(0, \frac{1}{2}, 1)$ valued solution to VLP on G, then (U, U') is a feasible solution to VP on G', where

$$u_p = \begin{cases} 1 & \text{if } \bar{u}_p = \frac{1}{2} \text{ or } 1 \\ 0 & \text{else} \end{cases}$$

and

$$u_p' = \begin{cases} 1 & \text{if } \bar{u}_p = 1 \\ 0 & \text{else.} \end{cases}$$

Furthermore $\bar{U}$ is optimal to VLP on G if and only if (U, U') is optimal to VP on G'.

Since the usefulness of the result of theorem (1.2.6) depends upon the number of variables having integer values in an optimal solution to VLP, they have also presented an algorithm to determine an optimal solution to VLP in which a maximal collection of variables are integer valued.

Finally using the above results Nemhauser and Trotter [43] have presented an algorithm for VP. An outline of this algorithm is : Given a vertex packing P, an exhaustive search is made on $\bar{V} = N/V$ for an augmenting subset I. If one is found, V is redefined as $(V \cup I)/V(I)$, and the procedure is repeated, if not, V is an optimal packing in G. To search an augmenting subset they have used the straight forward implicit enumeration scheme on $\bar{V}$. Computational results given by them show that this algorithm performs best on the graphs of low and high density. Graphs of medium density are of greater difficulty for the procedure, not only in terms of computational time, but also in terms of augmentations required to attain optimality. Picard and Queyranne [49] have proved that there is a unique maximal set of variables that are integer valued in an optimal solution to VLP. Using this, they concluded that the algorithm given by Nemhauser and Trotter [43] for VLP also gives an optimal solution having maximum number of integer valued nodes.

It looks that if C = e, for solving VP, it will be useful to solve VLP first. Pulleyblank [53] has shown that it is not so. He has defined a 2-bicritical graph which has the property that the unique optimal solution U^* to VLP defined

on such a graph is $u_p^* = \frac{1}{2}$ $\forall$ $p \in N$. He has also shown that almost all the graphs are 2-bicritical. Thus for $C = e$, VLP is 'almost always' of no use in solving VP for random graphs. He did not give any such comment for the case, when $C \neq e$.

Houck and Vemmungati [25] also studied vertex packing problem for $C = e$. They have also developed a branch and bound type of algorithm for solving this. They have also suggested a procedure to construct a good initial solution for VP which in some cases is known to be optimal. Since it has already been shown that the vertex packing on a bipartite graph can be solved efficiently, they obtain this initial vertex packing for G, by solving a vertex packing problem on a bipartite graph $\bar{G}$, which is generated from G by giving particular directions to its edges. They have proved that if the edges of G can be directed transitively, then the optimal solution thus obtained is an optimal solution to VP on G. They have also shown that the group theoretical approach to integer programming, when applied to vertex packing, yields a trivial group problem, regardless of the determinant of the associated basis. Using this, they have developed some nice fathoming results for their branch and bound type of algorithm for VP and have also shown that it is possible to permanently fix certain variables at either zero or one and consequently to reduce the graph G. Computational results on this algorithm also show that the problems with density .25 are harder than those with lower and higher densities.

Balas and Samuelsson [5] have also contributed significantly in this field. They restated the edge covering problem as a linear program (LPNC) whose constraints are the facets of the convex hull H_I of CP (LPNC) is formulated as :

$$\begin{array}{lll} \text{minimize} & \sum_{p \in N} u_p & \\ \text{such that} & \sum_{p \in Q_i} u_p \geq |Q_i| - 1 & \forall\, Q_i \in K \\ & \sum_{p} d_{h_p} u_p \geq d_{h_o} & h \in F \\ & u_p \geq 0, & p \in N \end{array}$$

where K is the family of all cliques of G. An inequality indexed by F is a facet of the convex hull, not associated with a clique. Dual formulation (LDNC) of (LPNC) is a relaxation of the edge matching problem MP, in the sense that a one-one correspondence can be established between the feasible solutions of LDNC and MP. They have developed a dual simplex type algorithm, introducing the notion of a dual node clique set, which allows identification of a partial cover associated with a dual feasible solution to LPNC. With the help of a partial cover, using a labelling method, primal infeasibility is gradually reduced, while integrality and dual feasibility are preserved. They have also reported that this method after minor modifications can be used to solve the weighted covering (of edges) problems. Till now very little computational experiences have been obtained to compare these existing algorithms.

1.3 GENERALIZED EDGE COVERING PROBLEM

Generalized edge covering problem (P) of this thesis is a generalization of the weighted edge covering problem (CP). In this problem, the notion of an edge cover is extended to allow any positive integer in the right hand side vector instead of the usual entry of ones for all the edges of G.

Mathematical formulation of the generalized edge covering problem will be as follow :

P :

$$\text{minimize} \quad \sum_{p \in N} c_p \, u_p$$

$$\text{such that} \quad u_p + u_q \geq r_{pq} \qquad \forall \, (p,q) \in E$$

$$u_p \geq 0, \text{Integer} \quad \forall \, p \in N.$$

1.4 SUMMARY OF THE THESIS

Motivation behind the present work is obtained by Nemhauser and Trotter's result for VP [43]. Besides the present chapter, this thesis contains five more chapters. Chapterwise summary of these are given below :

In the second chapter we show that every basic feasible solution of the generalized weighted edge covering problem (P), is (Integer, Integer / 2) valued. We also prove that all the nodes which got some integer values in an (Integer, Integer/2) valued optimal solution of LP (the linear relaxation of P) can retain the same values in an optimal solution of P. Thus we conclude that P can be decomposed into two problems :

(a) A linear programming problem LP,

(b) A weighted edge covering problem on a reduced graph of G.

Hence if a computationally good algorithm can be obtained to solve the weighted edge covering problem, it can effectively be utilized to solve this generalized edge covering problem too.

In the third chapter, we have shown that an (Integer, Integer/2) valued optimal solution of LP can be obtained by summing the optimal solutions of finite number of linear edge covering subproblems on the subgraphs G_i of G. Each such subproblem can easily be solved by solving a maximum flow problem on a bipartite graph generated by G_i. Maximum number of such subproblems are bounded by the highest value of the right hand side vector of P. The main result of this chapter, on which the above algorithm is based, is that, LP can be decomposed into two linear programming problems and the optimal solution of the LP is sum of the optimal solutions of these two decomposed problems.

In the fourth chapter, we have identified some more problems which can also be solved by the above algorithm, after some minor modifications. This will include generalized vertex packing problem, transportation problem and some other similar problems. A natural modification of the above algorithm is reported for the case when G is a bipartite graph.

An algorithm is also developed here, for the problem P, when the

underlying graph G is a tree. It is proved that in this case number of subproblems are bounded by kn^3, where k is a constant independent of r_{pq}'s and n is the number of nodes in G.

In the fifth chapter, computational results of the above algorithm for IP, using randomly generated test problems are reported. It is observed that the number of iterations are bounded by O(n).

Last chapter of this thesis is not directly related to the weighted edge covering problem, but deals with the "Construction of optimal communication spanning trees", first discussed by Hu [27]. In this chapter we give some algorithms to construct optimal communication spanning trees, having some special structures. Finally we suggest a minor modification to Hu's algorithm [27], which improves the computational efficiency of the algorithm.

CHAPTER II

DECOMPOSITION OF P[G;r]

2.1 INTRODUCTION

Generalized edge covering problem, defined in the previous chapter, is formulated as the following integer program :

P[G;r]

$$\begin{aligned} &\text{minimize} && CU \\ &\text{such that} && AU \geq r \\ & && U \geq 0, \text{ Integer} \end{aligned}$$

where C is an n-vector whose component c_p ($p = 1,\ldots,n$) is **a positive** integer which represents the cost of assigning unit weight to the node p of the graph G.

U is an n-vector of variables, whose component $u_p (p=1,\ldots,n)$ denotes the weight assigned to the node p.

r is an m-vector of requirements whose component r_{pq} represents the requirement for the edge (p,q) of G, $\forall\ (p,q) \in E$. Following is the linear relaxation of P[G;r],

LP[G;r]

$$\begin{aligned} &\text{minimize} && CU \\ &\text{such that} && AU \geq r \\ & && U \geq 0 \end{aligned}$$

In this chapter by exploiting the structure of P[G;r], we show that it is possible to decompose one of its optimal solution as the sum of following two vectors :

(a) A rounded down vector obtained from the (Integer, Integer/2) valued optimal solution of LP[G;r].

(b) An optimal solution of the weighted edge covering problem on a subgraph of G, generated from the optimal solution of LP[G;r].

Adding surplus variables U_s to LP[G;r], it can be written as :

$$\begin{aligned} \text{minimize} \quad & CU + OU_s \\ \text{such that} \quad & AU - IU_s = r \\ & U \geq 0 \\ & U_s \geq 0 \end{aligned} \qquad 2.1(a)$$

2.2 A RESULT ABOUT THE PROBLEM STRUCTURE

It is a well known result that in any basic feasible solution of the linear relaxation of the vertex packing problem, variables are $(0, \frac{1}{2}, 1)$ valued. Details of the proof of this result are available in [43]. Here we prove a similar result for LP[G;r], by a different approach.

Theorem 2.2.1 : Values of all the variables in any basic feasible solution of 2.1(a) are either integer or integer/2.

Proof : Let B be a basis of LP[G;r], when written in the form 2.1(a). After rearranging the columns, we can partition B as follows :

$$B = \left[\begin{array}{c|c} -I & A_1 \\ \hline 0 & A_2 \end{array}\right]$$

where I is an identity matrix and O is a null matrix.

For B to be nonsingular, it is necessary and sufficient that A_2 be nonsingular. Following two cases are possible for A_2 :

(a) A_2 is not an edge-node incidence matrix of a subgraph G_s of G.

(b) A_2 is an edge node incidence matrix of a subgraph G_s of G.

Case a : If A_2 is not an edge-node incidence matrix, it implies that there is at least one row of A_2 containing only one nonzero entry. In this case, by applying elementary operations, (multiplication by -1, addition and subtraction of rows) we can transform B to $\hat{B}$, such that $\hat{B} = R_1 B$

$$= \left[\begin{array}{c|c} -\hat{I} & \hat{A}_1 \\ \hline O & \hat{A}_2 \end{array}\right]$$

where R_1 is the product of elementary transformations of the type described above and which are used for transforming B to $\hat{B}$. $\hat{A}_2$ is a nonsingular edge-node incidence matrix of a subgraph G_s of G.

Case b : No transformation is needed and hence we take $R_1 = I$.

Since $\hat{A}_2$ is a nonsingular, edge-node incidence matrix of G_s, each component of G_s must have as many edges as nodes. Hence each component of G_s must be a cycle.

Again by renumbering of nodes, we can write

$$\hat{B} = \left[\begin{array}{c|cccc} \hat{P} & & \hat{\hat{A}}_1 & & \\ \hline O & Q_1 & & & \\ & & Q_2 & & \\ & & & \ddots & \\ & & & & Q_r \end{array}\right]$$

where Q_i is a cyclic matrix corresponding to the i^{th} component of G_s, $\forall\ i = 1,\ldots,t$; and $\hat{P}$ is an upper triangular matrix with entries 0,-1 and 1. $\hat{\hat{A}}_1$ corresponds to an acyclic portion of the graph. All the cycles of G_s will be of odd length, as $\hat{A}_2$ is non singular and determinant of an edge-node incidence matrix is 0 or $|2|$ according to as it corresponds to an even or odd cycle. Using elementary column operations of addition and subtraction on $\hat{B}$, we get

$$\hat{B}K = R_1BK = \left[\begin{array}{c|cccc} -\hat{\hat{I}} & & 0 & & \\ \hline 0 & T_1 & & & \\ & & T_2 & & \\ & & & \ddots & \\ & & & & T_t \end{array}\right]$$

where $\hat{\hat{I}}$ is an identity matrix and

$$T_i = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ 1 & -1 & 1 & -1 & \ldots & 2 \end{bmatrix}$$

Again using elementary column operations of addition and subtraction, we get

$$\Delta = RR_1BK = \left[\begin{array}{c|cccc} -\hat{I} & & 0 & & \\ \hline 0 & W_1 & & & \\ & & W_2 & & \\ & & & \ddots & \\ & & & & W_t \end{array}\right]$$

where,

$$R = \left[\begin{array}{c|cccc} \hat{I} & & 0 & & \\ \hline 0 & Z_1 & & & \\ & & Z_2 & & \\ & & & \ddots & \\ & & & & Z_t \end{array}\right]$$

and

$$Z_i = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ -1 & 1 & -1 & \ldots & 1 \end{bmatrix}$$

and $W_i = \begin{bmatrix} 1 & & & \\ & 1 & & 0 \\ & & \ddots & \\ 0 & & & 2 \end{bmatrix}$

Any basic solution of 2.1(a) is

$$B^{-1}r = (K\ \Delta^{-1}\ RR_1)\ r \qquad 2.2(a)$$

Since

$$W_i^{-1} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & \frac{1}{2} \end{bmatrix}$$

in 2.2(a), Δ^{-1} is the only matrix with some fractional elements and these are $\frac{1}{2}$'s only. Therefore each element of $B^{-1}r$ is either an integer or $\frac{1}{2}$ Integer.

Corollary 2.2.1 : If all the r_{pq}'s are nonnegative even numbers, every basic feasible solution of LP [G;r] will be a feasible solution of P [G;r] and hence any extreme point optimal solution of LP [G;r] will be an optimal solution of P [G;r].

2.3 SOME PROPERTIES OF THE SOLUTION OF LP [G;r]

Let U^t be an (Integer, Integer/2) valued optimal solution of LP [G;r]. Let $\bar{G}$ be a subgraph of G such that for any edge (p,q) of G,

$$u_p^t + u_q^t = r_{pq} \iff (p,q) \in \bar{G}$$

$\bar{G}$ may consist of one or more components. Let these components be denoted by $\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_{t_o}$. Following will hold good for each of these components :

Theorem 2.3.1 : For any j, $1 \leq j \leq t_o$, either all the u_p^t's where $p \in \bar{G}_j$ are integers or all are non integers.

Proof : Let p be a node of $\bar{G}_j$ and let u_p^t be non-integer. Since $c_p > 0$, there will exist at least one edge (p,q) such that $u_p^t + u_q^t = r_{pq}$. This shows that u_q^t is also a fraction. If there is any other node p' in $\bar{G}_j$, there will be a path from this node to p in $\bar{G}_j$. Again using the above argument, it is easy to show that the node p', as well as, all the nodes of the corresponding path will have noninteger values in U^t. Similarly it can also

be shown that if in a component $\bar{G}_j$, one node has integer value in U^t, then all the other nodes of this component will also have integer values in U^t.

Let $I = \{i_1, i_2, \ldots, i_{t_1}\}$ be the index set for the components of $\bar{G}$, whose nodes get integer values in U^t. Let $J = \{j_1, \ldots, j_{t_2}\}$ be the index set for the components of $\bar{G}$, whose nodes get noninteger values in U^t.

Let

$$\bar{\bar{G}}_j = \bar{G}_j \bigcup \{\text{edge } (p,q) \text{ of } G|\bar{G}, \text{ such that } p \text{ and } q \text{ both are in } \bar{G}_j\}.$$

Let $\bar{A}$ be an edge-node incidence matrix for the subgraph $\bar{G}$.

LP $[\bar{G};\bar{r}]$ can be written as :

$$\begin{aligned} &\text{minimize} && \bar{C}U \\ &\text{such that} && \bar{A}U \geq \bar{r} \\ & && U \geq 0 \end{aligned}$$

where $\bar{C}$ and $\bar{r}$ are subvectors of C and r corresponding to the subgraph $\bar{G}$.

From the definition of $\bar{G}$, it is obvious that U^t is an optimal solution for LP $[\bar{G}; \bar{r}]$ also. Decomposing LP $[\bar{G};\bar{r}]$ in terms of the components of $\bar{G}$ we get :

$$\text{LP}[\bar{G},\bar{r}] : \quad \text{minimize} \sum_{k \in I} \bar{C}_k U_k + \sum_{j \in J} \bar{C}_j U_j$$

such that,

$$\begin{bmatrix} \bar{A}_1 & & & \\ & \bar{A}_2 & & \\ & & \ddots & \\ & & & \bar{A}_{t_1+t_2} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{t_1+t_2} \end{bmatrix} \geq \begin{bmatrix} \bar{r}_1 \\ \bar{r}_2 \\ \vdots \\ \bar{r}_{t_1+t_2} \end{bmatrix}, U_i \geq 0 \quad \forall i = 1,2,\ldots t_1+t_2$$

where for every i ($i = 1,\ldots,t_1$) $\bar{A}_i$ is the edge-node incidence matrix $\bar{G}_i$ and for every j, ($j = 1,\ldots,t_2$), $\bar{A}_{t_1+j}$ is an edge-node incidence matrix of $\bar{G}_j$.

Therefore, restriction of U^t to $\bar{G}_i$, where $i \in I$, gives an optimal solution for LP $[\bar{G}_i; \bar{r}_i]$ and restriction of U^t to $\bar{G}_j$ where $j \in J$ is an optimal solution of LP $[\bar{G}_j; \bar{r}_j]$. We will denote these restrictions of U^t by X^i, for $i \in I$; and Y^j, for $j \in J$. From the construction of $\bar{\bar{G}}_j$, it is easy to see that for every $j \in J$, Y^j will be a feasible solution to LP $[\bar{\bar{G}}_j; \bar{\bar{r}}_j]$ and hence optimal also.

Following results exploit the structure of the constraint set 2.1(a) with respect to being able to decompose its right hand side vector.

<u>Observation 2.3.1</u> : Let Y_1 be an optimal linear solution of P1, where,

<u>P1</u>

$$\begin{aligned} &\text{minimize} && CY \\ &\text{such that} && AY \geq b \\ & && Y \geq 0 \\ &\text{and where} && b \geq 0 \end{aligned}$$

Let P1 be decomposed into two problems P2 and P3, where

P2 $\quad$ minimize $\quad CY$

$$\text{such that } AY \geq b_1$$
$$Y \geq 0$$

and

P3 $\quad$ minimize $\quad CY$

$$\text{such that } AY \geq b_2$$
$$Y \geq 0$$

in such a way that $b = b_1 + b_2$ and $b_1 \geq 0$ and $b_2 \geq 0$ and we can write $Y_1 = Y_2 + Y_3$ where Y_2 and Y_3 are feasible solutions to P2 and P3 respectively. Then Y_2 and Y_3 will be optimal solutions for P2 and P3 respectively.

The converse is of course not true. An example can easily be constructed to substantiate it.

Now we decompose Y^j as follows :

$$Y^j = V^j + \frac{1}{2} e \tag{2.3(a)}$$

where e is the summation vector. LP $[\bar{G}_j; \bar{r}_j]$ ($\forall\ j \in J$) can be decomposed, into the following two subproblems :

LP $[\bar{G}_j; \bar{r}_j - e]$

$$\text{minimize} \sum_{p \in \bar{G}_j} c_p u_p$$

$$\text{such that } u_p + u_q \geq r_{pq} - 1 \quad \forall\ (p,q) \in \bar{G}_j$$

$$u_p \geq 0 \quad \forall\ p \in \bar{G}_j$$

and

LP $[\bar{G}_j; e]$

$$\text{minimize} \sum_{p \in \bar{G}_j} c_p u_p$$

$$\text{such that } u_p + u_q \geq 1 \quad \forall (p,q) \in \bar{G}_j$$

$$u_p \geq 0 \quad \forall p \in \bar{G}_j$$

It is easy to see that V^j and $\frac{1}{2}$ e are feasible solutions to LP $[\bar{G}_j; \bar{r}_j - e]$ and LP $[\bar{G}_j; e]$ respectively. Therefore by observation (2.3.1) V^j and $\frac{1}{2}$ e will also be optimal solutions for their respective problems.

Now using the result that $\frac{1}{2}$ e is optimal for LP $[\bar{G}_j; e]$, we will prove the following :

Theorem 2.3.2 : At least one optimal solution of PA_1 will be an optimal solution for the problem PA_2 also, where

PA_1

$$\text{minimize} \sum_{p \in \bar{G}_j} c_p u_p$$

$$\text{such that } u_p + u_q \geq 1 \quad \forall (p,q) \in \bar{G}_j, \ j \in J$$

$$u_p, \text{ unrestricted integer } \forall p \in \bar{G}_j$$

PA_2

$$\text{minimize} \sum_{p \in \bar{G}_j} c_p u_p$$

$$\text{such that } u_p + u_q \geq 1 \quad \forall (p,q) \in \bar{G}_j$$

$$u_p = 0,1 \quad \forall p \in \bar{G}_j.$$

Proof : We will prove the above theorem by constructing an optimal solution of PA_2 from an optimal solution of PA_1.

Following can be stated for these problems :

(i) PA_1 is bounded below as the vector $U^j = \frac{1}{2}e$ is an optimal solution for its linear relaxation.

(ii) If in the optimal solution $\hat{U}$ of PA_1 all the variables are 0 or 1, then $\hat{U}$ is feasible for PA_2 and hence optimal.

(iii) In case, $\hat{U}$ is not feasible for PA_2 there will exist at least one node say p_1 of $\bar{G}_j$, such that $\hat{u}_{p_1} < 0$. This is so, because; suppose it is not true then at least one node say p_2 should have a value greater than one in it. In this case, there can not be any binding edge of PA_1, incident upon this node p_2 and therefore the objective value can be improved by reducing the value of this node by one. This will then contradict the optimality of $\hat{U}$ for PA_1. A portion of the graph $\bar{G}_j$ formed by the edges which are binding w.r.t. $\hat{U}$ and connected to the node p_1, may look like the figure given below :

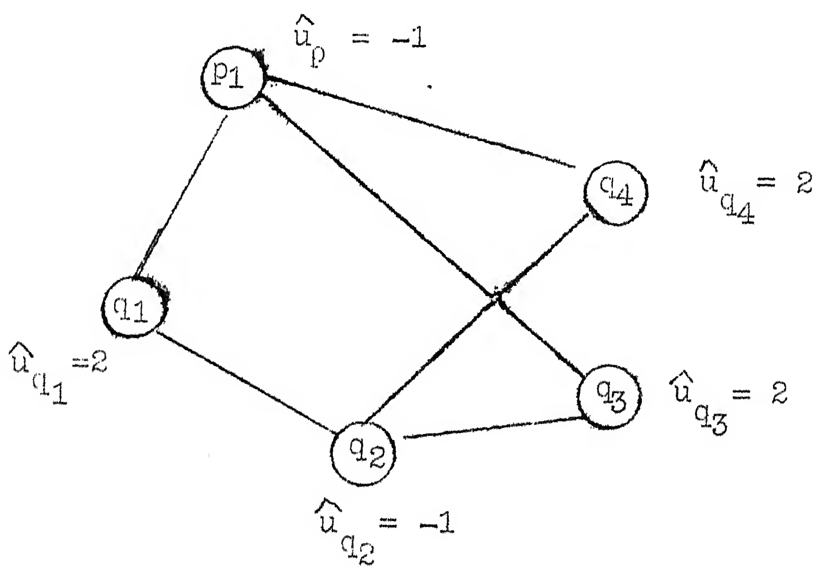


Fig. 2.1(1)

Construct the sets S_1 and E_1 as follows :

$S_1 = \{p \in G \mid \exists$ a path from p_1 to p, containing only the edges binding w.r.t. $\hat{U}\}$

$E_1 = \{(p,q) \in G \mid p$ and q both are in S_1 and (p,q) is a binding edge$\}$

Let $\hat{u}_p - \frac{1}{2} = d_p \qquad \forall\, p \in S_1$ 2.3(b)

Obviously d_p will be negative [positive] if and only if the corresponding component is negative [positive] in $\hat{U}$.

From the definition of S_1, it is easy to see that all its positive valued nodes will have equal value say 'a' and all its negative valued nodes will have the same value, say 'b'. Since $a = -b+1$, $b \leq -1$ and $\hat{U}$ is an integer solution, all the positive valued nodes of S_1, will have value greater than or equal to two in $\hat{U}$. Therefore we can write

$$d_p = \lambda \hat{d}_p \qquad \forall\, p \in S_1$$

where $\hat{d}_p \in \{1,-1\}$ and λ is a positive noninteger number greater than one. Having determined the structure of $\hat{U}$, we will now construct a solution of PA_2 which has the same objective value as $\hat{U}$. This will be done by constructing a sequence of optimal solutions for PA_1, starting with $\hat{U}$. For d_p (as defined above in 2.3(b)) following are the three possible cases :

either (a) $\sum_{p \in S_1} c_p\, d_p = 0$

or (b) $\sum_{p \in S_1} c_p d_p < 0$

or (c) $\sum_{p \in S_1} c_p d_p > 0$

We will now show that cases(b) and (c) are not possible. In case (a), we will be able to construct a vector $\hat{\hat{U}}$, which is also an optimal solution for PA_1 and in which either the number of negative valued nodes has decreased or the value of at least one negative valued node has increased, without decreasing the value of anyother negative valued node.

Case (a) : Let $\hat{\hat{U}}$ be defined as follows :

$$\hat{\hat{u}}_p = \tilde{u}_p - \hat{d}_p \quad \text{if } p \in S_1$$

$$\hat{\hat{u}}_p = \hat{u}_p \quad \text{otherwise.}$$

It is clear from the definition of d_p that the values of all the negative $\hat{u}_p$'s will go up by at least one. Therefore either

(a_1) : $\hat{\hat{U}}$ has less number of negative valued nodes, or

(a_2) : Values of some of the negative valued nodes of S_1 have gone up, while others have not gone down.

We will later prove that $\hat{\hat{U}}$ is also an optimal solution for PA_1. Hence for this case, proof of the theorem follows from the construction of a sequence of such optimal solutions ($\hat{U}$, $\hat{\hat{U}}$, $\hat{\hat{\hat{U}}}$,...), (where $\hat{\hat{\hat{U}}}$ is obtained from $\hat{\hat{U}}$ in the same manner as $\hat{\hat{U}}$ was constructed from $\ddot{U}$), for PA_1 till a feasible solution for PA_2 is obtained.

Case (b) : $\sum_{s \in S_1} c_p d_p < 0$

$$\implies \lambda \sum_{p \in S_1} c_p \hat{d}_p < 0$$

$$\implies \sum_{p \in S_1} c_p \hat{d}_p < 0 \qquad (\because \lambda > 0)$$

We now construct a vector U^* as follows :

$$u_p^* = \hat{u}_p + \hat{d}_p \qquad \text{if } p \in S_1$$

and $\quad u_p^* = \hat{u}_p \qquad$ otherwise.

It follows that

(1) $\sum_{p \in \bar{G}_j} c_p u_p^* < \sum_{p \in \bar{G}_j} c_p \hat{u}_p$

and

(2) U^* is a feasible solution for PA_1. (It will be proved later).

Thus U^* contradicts the optimality of $\hat{U}$ for PA_1. Hence case (b) can never occur for $\hat{U}$.

Case (c) : $\sum_{p \in S_1} c_p d_p > 0$

$$\implies \sum_{p \in S_1} c_p \hat{d}_p > 0$$

Construct a vector $\bar{U}$ as follows :

$$\bar{u}_p = \begin{cases} \hat{u}_p - \hat{d}_p & \text{if } p \in S_1 \\ \hat{u}_p & \text{otherwise.} \end{cases}$$

Therefore

$$\sum_{p \in \bar{G}_j} c_p \bar{u}_p < \sum_{p \in \bar{G}_j} c_p \hat{u}_p .$$

This inequality together with the feasibility of $\bar{U}$ for PA_1 (which will be proved later) contradicts the optimality of $\hat{U}$ for PA_1. Therefore case (c) is never possible for $\hat{U}$. Thus we have shown that for any optimal solution $\hat{U}$ of PA_1, if it is not feasible for PA_2, following condition must hold :

$$\sum_{p \in S_1} c_p d_p = 0 .$$

Thus it is always possible to construct another optimal solution for PA_1, which is less 'infeasible' for PA_2. Constructing a series of such optimal solutions of PA_1, an optimal solution of PA_2 with the same value of the objective function can always be constructed.

2.4 FEASIBILITY OF $\hat{\hat{U}}$, U* AND $\bar{U}$

For any edge (p,q) of $\bar{G}_j$, exactly one of the following is true :

(i) $(p,q) \in E_1$

(ii) $(p,q) \notin E_1$ and both the nodes p and q are in S_1.

(iii) $(p,q) \notin E_1$ and exactly one node from p and q is in S_1.

(iv) $(p,q) \notin E_1$ and neither p nor q is in S_1.

<u>Case (i)</u> : From the construction of the sets S_1 and E_1, we can say that one of the nodes p and q, say node p has a positive value

in $\hat{U}$ and the other node i.e. q has negative value in $\hat{U}$. Therefore $\hat{d}_p$ will be +1 and $\hat{d}_q$ will be -1. In $\hat{\hat{U}}$ and $\bar{U}$, value of this node p is decreased by one, while value of the node q is increased by one. Thus $\hat{\hat{u}}_p + \hat{\hat{u}}_q = 1$, since $(p,q) \in E_1$ and $\bar{u}_p + \bar{u}_q = 1$ since $(p,q) \in E_1$. In U^*, value of node p is increased by one, while the value of node q is decreased by one. Hence

$$u_p^* + u_q^* = 1 \quad \text{since } (p,q) \in E_1.$$

Case (ii) : $(p,q) \notin E_1$, this implies that there is a positive surplus on the edge (p,q) in $\hat{U}$. Since p and q both are in S_1, it is necessary that both must have positive values in $\hat{U}$. All the positive valued nodes of S_1 have an equal value and all the negative valued nodes of S_1 have an equal value. Surplus on any such edge (p,q) will be ≥ 3, because the positive valued nodes are greater than or equal to two. Therefore $u_p + u_q \geq 1$ will be satisfied by $\hat{\hat{U}}$, $\bar{U}$ and U^*.

Case (iii) : Let $p \in S_1$. There is a positive surplus greater than or equal to one on this edge (p,q), for the solution $\hat{U}$. Now there are two cases :

(a) $\hat{u}_p$ is negative and $\hat{u}_q$ is positive.

(b) $\hat{u}_p$ is positive and $\hat{u}_q$ is either positive or negative.

Case (a) : In $\hat{\hat{U}}$ and $\bar{U}$ value of the p^{th} node is increased by one, while the value of node q is unchanged. Hence

$$\hat{\hat{u}}_p + \hat{\hat{u}}_q \geq 1$$

and $\bar{u}_p + \bar{u}_q \geq 1$

In U*, value of node p is reduced by one, while the value of node q is unchanged. Hence we still have

$$u^*_p + u^*_q \geq 1.$$

Case (b) : Similar to case (iii) (a).

Case (iv) : In $\hat{U}$, U* and $\bar{U}$, values of the nodes p and q are unchanged. Hence the constraint $u_p + u_q \geq 1$ is satisfied by all of them.

2.5 DECOMPOSITION OF THE OPTIMAL SOLUTION

Using the above results, we will now prove the following theorem :

Theorem 2.5.1 : $(V^j + O^j)$ is an optimal solution of $P[\bar{G}_j;\bar{r}_j]$ $(\forall\ j \in J)$, where O^j is an optimal weighted edge covering for $\bar{G}_j$ and V^j is, as defined in section 2.3.

Proof : Let W^j be an optimal solution for $P[\bar{G}_j;\bar{r}_j]$ and let $d = W^j - Y^j$ where Y^j is an optimal solution for $LP[\bar{G}_j;\bar{r}_j]$ as defined in (2.3). Y^j can be written as

$$Y^j = V^j + \frac{1}{2}\, e \qquad 2.5(a)$$

All the half spaces $(u_p + u_q \geq 1)$ $(\forall (p,q) \in \bar{G}_j)$ passing through the point $\frac{1}{2}$ e, generate a cone Z_1 (with vertex $\frac{1}{2}$ e), which is the solution set for $LP[\bar{G}_j;e]$ without nonnegativity restrictions imposed on its variables. All the half spaces $u_p + u_q \geq r_{pq}$ $(\forall\ (p,q) \in \bar{G}_j)$ passing through the point Y^j generate a cone Z_2

with vertex at Y^j. Z_2 is the solution set for LP $[\bar{G}_j;\bar{r}_j]$ when nonnegativity restrictions are ignored. Since $r_{pq} \geq 1 \; \forall (p,q) \in \bar{G}_j$, Z_2 is contained in Z_1. Hence $d + \frac{1}{2}e$ is contained in Z_1 and is also integer valued. This may or may not be feasible for LP $[\bar{G}_j;e]$, but is definitely feasible for PA_1.

By theorem 2.3.1

$$\sum_{p \in \bar{G}_j} (d_p + \tfrac{1}{2}) \geq \sum_{p \in \bar{G}_j} c_p \, o_p^j$$

$$\Longrightarrow \sum_{p \in \bar{G}_j} c_p \, d_p \geq \sum_{p \in \bar{G}_j} c_p \, (o_p^j - \tfrac{1}{2}) \qquad 2.5(b)$$

Now we want to show that W^j is no better than $V^j + O^j$.

Let

$$\sum_{p \in \bar{G}_j} c_p \, w_p^j < \sum_{p \in \bar{G}_j} c_p \, (v_p^j + o_p^j)$$

$$\Longrightarrow \sum_{p \in \bar{G}_j} c_p (y_p^j + d_p) < \sum_{p \in \bar{G}_j} (v_p^j + o_p^j)$$

$$\Longrightarrow \sum_{p \in \bar{G}_j} c_p \, d_p < \sum_{p \in \bar{G}_j} (o_p^j - \tfrac{1}{2}),$$

which contradicts the inequality 2.5(b). Hence $V^j + O^j$ is an optimal solution to P $[\bar{G}_j;\bar{r}_j]$.

It is obvious that

$$U^* = \Big(\bigcup_{i \in I} X^i, \; \bigcup_{j \in J} (V^j + O^j)\Big)$$

is an optimal solution for P $[\bar{G};\bar{r}]$.

It is also easy to see that the requirements of all the other constraints corresponding to the edges which are not present in $[\bar{G};\bar{r}]$ are met by U^*. Hence U^* is an optimal solution for $P[G;r]$.

Thus we have reduced solving this integer programming problem $P[G;r]$, into two parts. In the first part, we solve a linear programming problem, which is a linear relaxation of the given integer problem and in the second part, we solve a weighted edge covering problem.

Since the optimal linear solution would quite often reduce the graph G into several components, above weighted edge covering problem can be solved independently on each subgraph. Hence any efficient algorithm [5, 25, 36, 43] for the weighted edge covering problem can be used to solve this generalized edge covering problem too.

CHAPTER III

ALGORITHM FOR LP $[G;r]$

3.1 INTRODUCTION

In this chapter we discuss the problem LP $[G;r]$ which is defined in chapter two. Although Simplex method can be used for solving it, we here present another exact algorithm, which is also iterative in nature. This algorithm is specially designed for exploiting the structure of the constraint set of LP $[G;r]$. In each iteration we solve only a maximum flow problem on some symmetric bipartite graph, generated by a subgraph of G.

3.2 NOTATIONS

For the i^{th} iteration where $i \geq 1$ we define

$\max 1(i) = \max_{(p,q) \in G} \{r_{pq}(i-1)\}$ where, $r_{pq}(i-1)$ is a component of the updated requirement vector $r(i-1)$ defined in section 3.3.

$\max 2(i) = \max_{(p,q) \in G} \{r_{pq}(i-1) \mid r_{pq}(i-1) \neq \max 1\ (i)\}$

$r_i \quad = \max 1(i) - \max 2(i)$

$s_i \quad = \lceil \max 1(i) - \max 2(i) \rceil$

where $\lceil \max 1(i) - \max 2(i) \rceil$ is the smallest integer greater than or equal to $(\max 1(i) - \max 2(i))$

$N^G(p) \quad = \{q \mid \exists\ (p,q) \in G\}$ i.e., it is a set of all the neighbouring nodes of p in G.

G_i : A subgraph of G generated for the i^{th} iteration, consisting of all those edges of G, for which the requirement updated after $(i-1)^{th}$ iteration, is max 1(i).

$\bar{G}(G_i)$: A bipartite graph generated by G_i, whose node set is $N_1 \cup N_2$, where $N_1 = N_2$ and N_1 is the node set of G_i, and $(p,q) \in \bar{G}(G_i)$ if and only if $(p,q) \in G_i$.

r(i) : Requirement vector, updated after i^{th} iteration. Formula for updating r(i) is given in 3.3.

r(0) : Given requirement vector.

3.3 OUT LINE OF THE ALGORITHM AND SOME RELATED RESULTS :

Algorithm 3.3

Let U(1) be an optimal solution of IP $[G_1; r_1 e]$ for the first iteration. We update the requirement vector r(0) with respect to the solution U(1). r(1), the requirement vector, updated after first iteration, is obtained as follows :

$$r_{pq}(1) = r_{pq}(0) - u_p(1) - u_q(1) \quad \text{if } r_{pq}(0) \geq u_p(1) + u_q(1)$$
$$= 0 \quad \text{otherwise.}$$

We now work with IP $[G; r(1)]$ and repeat the whole procedure till all the requirements have been met. At any intermediate iteration i, we calculate r_i and find G_i from the requirement vector r(i-1). Solution of IP $[G_i; r_i\, e]$ can be obtained by first solving the problem IP $[G_i; e]$ and then multiplying it

by r_i. Optimal solution of LP $[G_i;e]$ will be obtained by solving a maximum flow problem on $\bar{G}(G_i)$. Proof of this is based on the results obtained by Nemhauser and Trotter [43]. By using the transformation, (described in chapter one) for reducing a vertex packing problem to an edge covering problem, the theorem by Nemhauser and Trotter [43] can be adopted for an edge covering problem in the following form:

Theorem 3.3.1 : (i) If (U^1,U^2) is a feasible solution of $P[\bar{G}(G_i);e]$, then $\bar{U} = \frac{U^1+U^2}{2}$ is feasible $(0, \frac{1}{2}, 1)$ valued solution to LP $[G_i;e]$.

(ii) If $\bar{U}$ is a feasible $(0, \frac{1}{2}, 1)$ valued solution to LP $[G_i;e]$ then (U^1,U^2) is a feasible solution to $P[\bar{G}(G_i);e]$ where,

$$u_j^1 = \begin{cases} 1 & \text{if } \bar{u}_j = \frac{1}{2} \text{ or } 1 \\ 0 & \text{otherwise} \end{cases}$$

and
$$u_j^2 = \begin{cases} 1 & \text{if } \bar{u}_j = 1 \\ 0 & \text{otherwise} \end{cases}$$

Furthermore $\bar{U}$ is an optimal solution to LP $[G_i;e]$ if and only if (U^1,U^2) is optimal to $P[\bar{G}(G_i);e]$.

After establishing this one-one correspondence between solutions of LP $[G_i;e]$ and $P[\bar{G}(G_i);e]$, we obtain the optimal solution of $P[\bar{G}(G_i);e]$ by solving the dual of LP $[\bar{G}(G_i);e]$ which is a maximum flow problem on $\bar{G}(G_i)$.

Dual of LP $[\bar{G}(G_i);e]$ is :

DLP $[\bar{G}(G_i);\ e]$

$$\text{maximize} \sum_{(p,q)\,\in\,\bar{G}(G_i)} x_{pq}$$

$$\text{such that} \quad \sum_{q \in N_2} x_{pq} \leq c_p \quad \forall\, p \in N_1$$

$$\sum_{p \in N_1} x_{pq} \leq c_q \quad \forall\, q \in N_2$$

$$x_{pq} \geq 0 \quad \forall (p,q) \in \bar{G}(G_i).$$

Let $\bar{X}$ be an optimal solution for DLP $[\bar{G}(G_i);e]$. It is shown in Appendix A that if $\bar{X}$ is not a basic solution, even then, we can directly construct a (0,1) valued feasible solution (U^1,U^2) of LP $[\bar{G}(G_i);e]$ which satisfies all the complementary slackness conditions with respect to $\bar{X}$, and hence the solution thus obtained will be an optimal solution for LP $[\bar{G}(G_i);e]$. Thus a $(0, \frac{1}{2}, 1)$ valued optimal solution of LP $[G_i;e]$ is obtained. Let U(i) denote an optimal solution of LP $[G_i;r_i\ e]$. Here $U(i) = r_i\,\bar{U}$, where $\bar{U}$ is solution of LP $[G_i;e]$ and is available in terms of (U^1,U^2). We will prove that U^* is an optimal solution where for some k

$$U^* = U(1) + U(2) + \ldots + U(k)$$

We will consider the following possible cases :

(1) All r_i's are integers

(2) Some r_i's are integers and some are non integers.

We will first show that the proposed algorithm gives an optimal solution for case (1), in a finite number of iterations.

Outline of the proof for Optimality and Finiteness :

r_{pq}'s are given to be positive integers, and at every iteration i, some of them are being updated by 'r_i' which is also a positive integer. Therefore all the requirements would be met after a finite number of iterations. Let k be the total number of iterations needed. We consider now the following two problems :

Problem (A) : Subproblem being solved at the i^{th} iteration, i.e.

LP $[G_i; r_i\, e]$

$$\text{minimize} \sum_{p \in G_i} c_p u_p$$

$$\text{such that } u_p + u_q \geq r_i, \quad \forall (p,q) \in G_i$$

$$u_p \geq 0, \quad \forall p \in G_i$$

Problem (B) : The linear programming problem, with the right hand side vector denoting the updated requirements after i^{th} iteration. This problem will be denoted by

LP $[G; r(i)]$:

$$\text{minimize} \sum_{p \in G} c_p u_p$$

$$\text{such that } u_p + u_q \geq r_{pq}(i) \quad \forall (p,q) \in G$$

$$u_p \geq 0, \quad \forall p \in G$$

Let $U^B(i) = U(i+1) + U(i+2) + \dots + U(k)$ 3.3(a)

Since $U^B(i-1) = U(i) + U^B(i)$

$U^B(0) = U^*.$

It can easily be verified that $U^B(i)$ is a feasible solution for LP $[G;r(i)]$. $U^B(i) + U(i)$ will be shown to be an optimal solution of LP $[G;r(i-1)]$, $\forall\ i = 1,2,\ldots,k$. Let G' be a subgraph of G containing all the nodes of G and all its edges which are binding with respect to U*.

For a given i, where $1 \leq i \leq k$, let

$$S = \{p \mid u_p(i) \geq 0, \text{ but } u_p^B(i) = 0\}$$

We now prove some of the properties that are satisfied by U(i) and $U^B(i)$.

(P1) : If $(p,q) \in G_i$ and it is binding with respect to U(i) i.e.,

$$u_p(i) + u_q(i) = r_i$$

then the updated requirement on this edge will be the highest. In other words,

$$r_{pq}(i) = \max 1\ (i + 1).$$

<u>Proof</u> : For any edge (p',q') following three cases are possible :

(a) $(p',q') \notin G_i$

(b) $(p',q') \in G_i$ and it is binding w.r.t. U(i)

(c) $(p',q') \in G_i$ and it is a non-binding edge w.r.t. U(i).

<u>Case (a)</u> : $(p',q') \notin G_i \Longrightarrow r_{p'q'}(i-1) \leq \max 2(i)$. Since $u_{p'}(i)$ and $u_{q'}(i) \geq 0$

$$r_{p'q'}(i) \leq \max 2(i).$$

<u>Case (b)</u> : $(p',q') \in G_i \Longrightarrow r_{p'q'}(i-1) = \max 1(i)$ and since (p',q') is binding w.r.t. U(i),

$$\therefore\ r_{p'q'}(i) = \max\{r_{p'q'}(i-1) - u_{p'}(i) - u_{q'}(i), 0\}$$

$$= \max\{\max 1(i) - \max 1(i) + \max 2(i), 0\}$$

$$= \max 2(i) \qquad \because \qquad \max 2(i) \geq 0.$$

Case (c) : $(p',q') \in G_i$ and let $u_{p'}(i) + u_{q'}(i) = r_i + \epsilon$ for some $\epsilon > 0$

Since $r_{p'q'}(i) = \max\{r_{p'q'}(i-1) - u_{p'}(i) - u_{q'}(i), 0\}$

$$r_{p'q'}(i) = \max\{\max 1(i) - r_i - \epsilon, 0\}$$

$$= \max\{\max 2(i) - \epsilon, 0\}$$

$$< \max 2(i).$$

Hence we conclude that

$$\max 1(i+1) = \max 2(i).$$

(P2) : For every iteration i, all the edges of G corresponding to the positive components of $r(i)$ and which are also binding with respect to $U^B(i)$, will also be present in G'.

Proof : Let (p,q) be such an edge. By definition

$$r_{pq}(i) = r_{pq} - \sum_{j=1}^{i} (u_p(j) + u_q(j))$$

Therefore if (p,q) is a binding edge of $U^B(i)$ and $r_{pq}(i) > 0$, then

$$u_p^B(i) + u_q^B(i) + \sum_{j=1}^{i-1} (u_p(j) + u_q(j)) = r_{pq}.$$

But by equation 3.3(a), left hand side of this equation is $u_p^* + u_q^*$. Hence $(p,q) \in G'$.

(P3) : If a node p is in the set S, then all the edges of $LP[G_i; r_i\, e]$ which are incident upon p and which are also binding with respect to U(i) will be present in G'.

Proof: Let $(p,p_1), (p,p_2), \ldots, (p,p_s)$ be all the binding edges of $LP[G_i; r_i\, e]$, incident upon $p \in S$.

In proving (P1), we have shown that

$$\max 1(i+1) = \max 2(i) \quad \text{for any } i \text{ such that } 1 \leq i \leq k.$$

Therefore we can write

$$\max 1(i+1) = r_{i+1} + r_{i+2} + \ldots + r_k \qquad 3.3(b)$$

Since $U^B(i)$ is feasible for $LP[G; r(i)]$ it will satisfy the following constraints

$$u_p^B(i) + u_{p_j}^B(i) \geq r_{pp_j}(i) \qquad \forall\, j = 1,\ldots,s$$

But by property P1, $r_{pp_j}(i) = \max 1(i+1)$ $\forall\, j = 1,2,\ldots,s$ and since $p \in S$, $u_p^B(i) = 0$, therefore $u_{p_j}^B(i) \geq \max 1(i+1)$ $\forall\, j = 1,\ldots,s$. We will now show that $u_{p_j}^B(i) = \max 1(i+1)$ $\forall\, j = 1,2,\ldots,s$. Let $u_{p_{j_o}}^B(i) > \max 1(i+1)$ for some $j_o, 1 \leq j_o \leq s$, then by 3.3(b), there exists at least one t, $(i+1 \leq t \leq k)$, such that

$$u_{p_{j_o}}(t) > r_t \qquad 3.3(c)$$

Since $c_{p_{j_o}} > 0$, it will be possible to reduce the value of CU(t). But this will contradict the optimality of U(t) for $LP[G_t; r_t\, e]$.

Hence $u^B_{p_j}(i) = \max 1(i+1) \quad \forall\ j = 1,2,\ldots,s$. Now

$$u^*_p + u^*_{p_j} = \left(\sum_{z=1}^{i} u_p(z) + \sum_{z=1}^{i} u_{p_j}(z)\right) + \left(u^B_p(i) + u^B_{p_j}(i)\right)$$

$$= (r_{pp_j} - \max 1(i+1)) + \max 1(i+1)$$

$$= r_{pp_j}$$

Hence $(p,p_j) \in G' \quad \forall \quad j = 1,2,\ldots,s$.

(P4) : Let p_1 be a node adjacent to a node $p \in S$, such that (p_1,p) is an edge of G_i, and binding with respect to $U(i)$, then no node adjacent to p_1 in G' can take a positive value in $U^B(i)$.

Proof : Let $(p_1,q_1) \in G'$. Since (p,p_1) is a binding edge with respect to $U(i)$, by property P1.

$$r_{pp_1}(i) = \max 1(i+1)$$

As $p \in S$, by property (P3),

$$u^B_{p_1}(i) = \max 1(i+1).$$

We will now show that $u^B_{q_1}(i) = 0$. Let $u^B_{q_1}(i) > 0$

$$\therefore\ u^B_{p_1}(i) + u^B_{q_1}(i) > \max 1(i+1) \qquad 3.3(d)$$

Since $r_{p_1q_1} - \sum_{j=1}^{i} (u_{p_1}(j) + u_{q_1}(j)) \leq \max 1(i+1)$.

Therefore

$$r_{p_1q_1} - (u^*_{p_1} + u^*_{q_1}) \leq \max 1(i+1) - (u^B_{p_1}(i) + u^B_{q_1}(i)).$$

But by 3.3(d), right hand side of the above inequality is a negative number,

$$\therefore \quad r_{p_1 q_1} < u^*_{p_1} + u^*_{q_1}$$

This contradicts the assumption that $(p_1, q_1) \in G'$.

Hence $u^B_{q_1}(i) = 0$.

(P5) : Let $\{q_1, q_2, \ldots, q_t\}$ be a subset of positive valued nodes in U(i). If we get a feasible solution of LP $[G_i; r_i\ e]$, from U(i), by reducing the values of each of the nodes $q_1, q_2, \ldots, q_t$ by $\frac{1}{2}$ and increasing by $\frac{1}{2}$, only the values of nodes in $B(q_1, q_2, \ldots, q_t)$, where

$$B(q_1, q_2, \ldots, q_t) = \{p \mid (p, q_j) \text{ is a binding edge w.r.t } U(i) \text{ for some } j = 1,2,\ldots,t\}$$

then $c_{q_1} + \ldots + c_{q_t} \leq \sum_{p \in B(q_1, \ldots, q_t)} c_p$.

Proof : Follows immediately from the optimality of U(i) for LP $[G_i; r_i\ e]$.

Theorem 3.3.3 : $U(i) + U^B(i)$ is an optimal solution to LP $[G'; r'(i-1)]$ if $U^B(i)$ is an optimal solution to LP $[G'; r'(i)]$, where $r'(i)$ is a subvector of $r(i)$ corresponding to the edge set of G'.

Proof : It will be proved by contradiction. Let Y be an optimal solution of LP $[G', r'(i-1)]$ such that

$$CY > C(U(i) + U^B(i)).$$

Since $c_p > 0$; $\forall\, p \in G'$, there will exist at least one node p_o, for which

$$u_{p_o}(i) + u^B_{p_o}(i) > y_{p_o}.$$

Since Y is an optimal solution of a linear programming problem, we can select Y to be a basic solution, therefore by theorem (2.2.1), Y will have integer, integer/2 values for its components.

We partition the node set of G', into the following subsets :

$S_1^D = \{p \mid p \in S \text{ and } y_p < u_p(i) + u_p^B(i)\}$

$S_1^I = \{p \mid p$ is connected to a node of S_1^D by a binding edge of $LP[G_i; r_i\, e]$ and $y_p > u_p(i) + u_p^B(i)\}$

$S_2^D = \{p \mid u_p^B(i) > 0 \text{ and } y_p < u_p(i) + u_p^B(i)\}$

$S_2^I = \{p \mid p$ is connected to a node of S_2^D by a binding edge of problem $LP[G'; r(i)]$ and $y_p > u_p(i) + u_p^B(i)\}$

$S_1 = \{p \mid p \notin S_1^D \cup S_2^D \cup S_1^I \cup S_2^I, \text{ and } y_p > u_p(i) + u_p^B(i)\}$

$S_2 = \{p \mid y_p = u_p(i) + u_p^B(i)\}$

Thus $\forall\, p \in S_1^D \cup S_2^D$,

$$y_p < u_p(i) + u_p^B(i),$$

and

$$\forall\, p \in S_1^I \cup S_2^I \cup S_1$$

$$y_p > u_p(i) + u_p^B(i).$$

We now prove some properties of these sets :

(a) $S_1^D \bigcap S_2^D = \phi$

$S_i^D \bigcap S_j^I = \phi \quad \forall \; i,j \in \{1,2\}$

These follow by the definitions of S_1^D, S_1^I, S_2^D and S_2^I.

(b) $S_1^I \bigcap S_2^I = \phi$

Proof is by contradiction.

Let $S_1^I \bigcap S_2^I \neq \phi$, and let $p_o \in S_1^I \bigcap S_2^I$

$p_o \in S_1^I \implies p_o$ is connected to a node of S, by a binding edge of LP$[G_i;r_i\ e]$.

$p_o \in S_2^I \implies p_o$ should be adjacent to a node q_o in G' such that $u_{q_o}^B(i) > 0$. But by property (P4), no adjacent node of p_o in G' can be positive valued in $U^B(i)$, therefore $S_2^I \bigcap S_2^I = \phi$.

(c) It is easy to see that every constraint of LP$[G',r'(i-1)]$ is binding with respect to $U(i) + U^B(i)$. Therefore, there will be no edge of G' which connects two nodes of $S_1^I \bigcup S_2^I \bigcup S_1$ and is also binding with respect to Y.

Similarly there can not be any edge (p_1,q_1) of G', such that $p_1 \in S_1^I \bigcup S_2^I \bigcup S_1$ and $q_1 \in S_2$.

(d) We will now show that

(i) $- \sum_{p \in S_1^D} c_p + \sum_{q \in S_1^I} c_q \geq 0$

and(ii) $- \sum_{p \in S_2^D} c_p + \sum_{q \in S_2^I} c_q \geq 0$.

<u>Case (i)</u> : $p \in S_1^D \implies p \in S$, therefore by property (P3), every edge (p,q) which is binding with respect to U(i), will be

present in G' and therefore it will be binding in LP[G,r(i)] also. Since

$$y_p < u_p(i) + u_p^B(i)$$

and $y_p + y_q \geq r_{pq}(i-1)$

$$= u_p(i) + u_q(i) + u_p^B(i) + u_q^B(i)$$

therefore $u_q > u_q(i) + u_q^B(i)$.

Hence $\quad q \in S_1^I$.

Thus we can say that for every node $p \in S_1^D$, all the neighbouring nodes q_j of G_j, such that (p,q_j) is binding with respect to U(i), will be present in S_1^I.

Hence by property (P5)

$$-\sum_{p \in S_1^D} c_p + \sum_{q \in S_1^I} c_q \geq 0$$

<u>Case (ii)</u> : $\quad p_1 \in S_2^D \implies u_{p_1}^B(i) > 0$.

By property (P2), all the edges, incident upon p_1 and binding with respect to $U^B(i)$ are present in G'. Let (p_1,q_1) be one such edge. As in case (i) we can show that $q_1 \in S_2^I$. Since $U^B(i)$ is assumed to be an optimal solution of LP[G';r'(i)]

$$-\sum_{p \in S_2^D} c_p + \sum_{q \in S_2^I} c_q \geq 0$$

Hence, we can say that

$$\sum_{p \in S_1^D \cup S_2^D} c_p - \sum_{q \in S_1^I \cup S_2^I} c_q \leq 0.$$

Let us now construct a vector $\bar{Y}$, from Y by making the following additions and subtractions :

(1) Decrease the y_p's by a constant d for all the nodes p of $S_1^I \cup S_2^I$ where $d \leq \frac{1}{4}$.

(2) Increase the y_p's by d for all the nodes p of $S_1^D \cup S_2^D$.

By property (P3), it is easy to see that $\bar{Y}$ is a feasible solution of LP [G';r(i-1)] and

$$-CY + C\bar{Y} = d \left[\sum_{p \in S_1^D \cup S_2^D} c_p - \sum_{q \in S_1^I \cup S_2^I} c_q - \sum_{s \in S_1} c_s \right]$$

Since the right hand side of the above equality is a negative number, a contradiction to the optimality of Y, is obtained. Hence $U(i) + U^B(i)$ is an optimal solution to LP [G';r'(i-1)], provided $U^B(i)$ is an optimal solution for LP [G';r'(i)]. □

<u>Theorem 3.3.4</u> : $U^* = U(1) + U(2) + \ldots + U(k)$ is an optimal solution for LP [G;r].

<u>Proof</u> : We will first show that U* is an optimal solution to LP [G';r'], where r' is a subvector of r, corresponding to the edges of G'. We will prove this theorem recursively.

<u>Step 1</u> : <u>$U^B(k-1)$ is an optimal solution for LP [G';r'(k-1)]</u> :

Since k is the last iteration, all the left over requirements would be met by U(k).

Problem solved at the k^{th} iteration is

$$\text{minimize} \sum_{p \in G_k} c_p u_p$$

$$\text{such that } u_p + u_q \geq \max 1(k) \quad \forall (p,q) \in G_k$$

$$u_p \geq 0 \qquad \forall p \in G_k$$

Since all the edges of this problem, which are binding with respect to U(k), are present in G', U(k) is also optimal for LP [G';r'(k-1)]. Hence $U^B(k-1) = U(k)$ is optimal for LP [G';r'(k-1)].

Step 2 : By theorem (3.3.3), $U^B(k-2) = U(k-1) + U^B(k-1)$ is optimal for LP [G';r'(k-2)], because $U^B(k-1)$ is optimal for LP [G';r'(k-1)].

Step 3 : Let $U^B(k-i)$ be an optimal solution for LP [G';r'(k-i)]. By theorem (3.3.3) $U^B(k-i-1) = U(k-i) + U^B(k-i)$ is optimal for LP [G';r'(k-i-1)]. Hence we conclude that $U^B(0) = U^*$ is an optimal solution for LP [G';r'].

From the construction of LP [G;r(i)] for i = 1,2,...,k, it is easy to see that U^* is a feasible solution to LP [G;r] also. Since LP [G';r'] has fewer constraints, U^* is also an optimal solution for LP [G;r].

□

3.4 NONFINITENESS OF THE PROPOSED ALGORITHM, WHEN SOME r_i's ARE FRACTIONS

It is easy to see that for any iteration i of the above proposed algorithm where $(2 \leq i \leq k)$, number of nodes in G_i is greater than or equal to the number of nodes in G_{i-1}, but it is quite possible that the number of iterations with the same set of nodes may not be finite if some r_i's become fractions. For example consider the LP problem on the following graph :

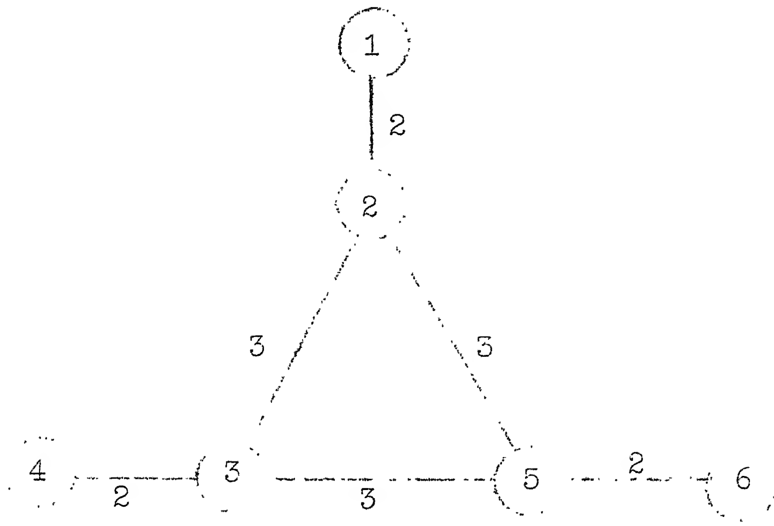


Fig. 3.4(1)

(Number beside an edge represents its requirement)

Let $c_p = 1 \;\forall\, p = 1,2,\ldots,6$.

Iteration 1 : max 1(1) = 3

max 2(1) = 2

$r_1 = 1$

Therefore G_1 is

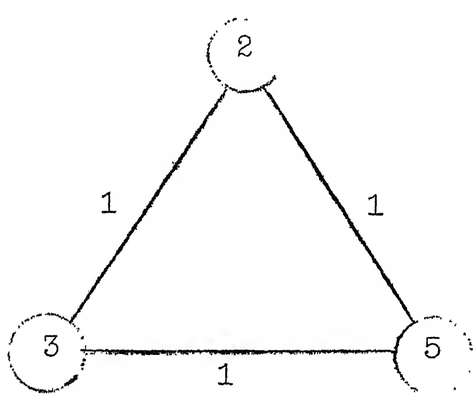


Fig. 3.4(2)

An optimal solution for LP $[G_1;e]$ is

$$U(1) = \begin{bmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix}$$

and, $[G; r(1)]$

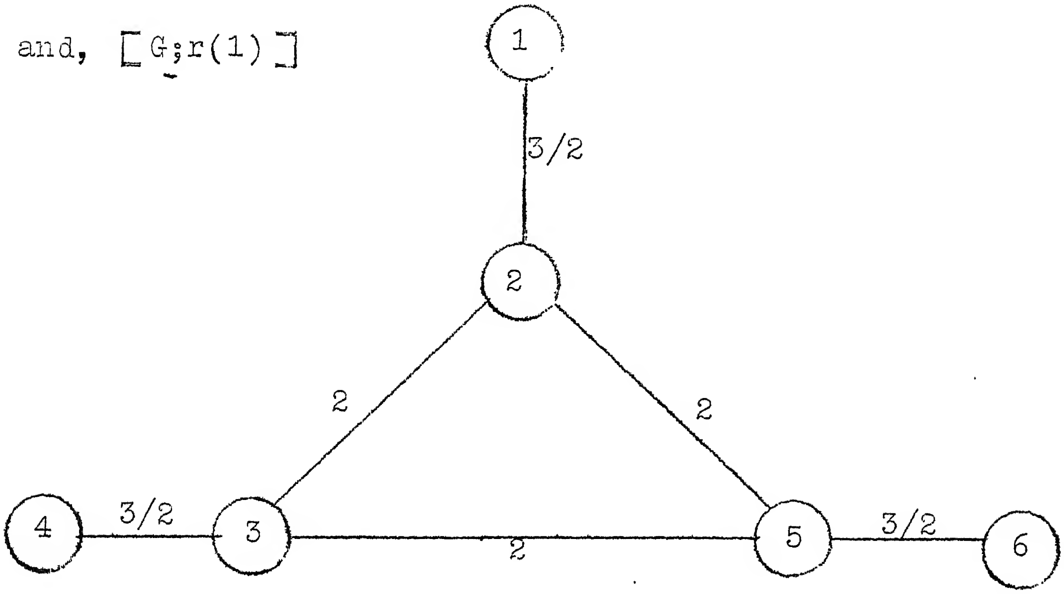


Fig. 3.4(3)

Iteration 2 :

$$\max 1(2) = 2$$

$$\max 2(2) = 3/2$$

$$r_2 = \frac{1}{2}$$

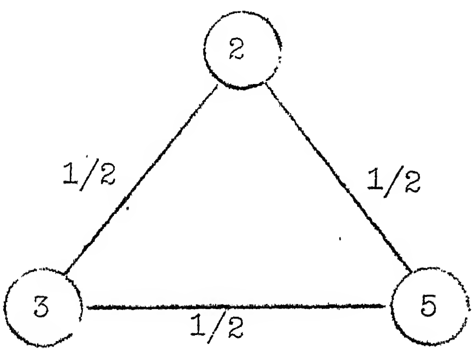


Fig. 3.4(4)

An optimal solution for LP $[G_2; r_2\, e]$ is

$$U(2) = \begin{bmatrix} 0 \\ \frac{1}{2^2} \\ \frac{1}{2^2} \\ 0 \\ \frac{1}{2^2} \\ 0 \end{bmatrix}$$

and $[G; r(2)]$

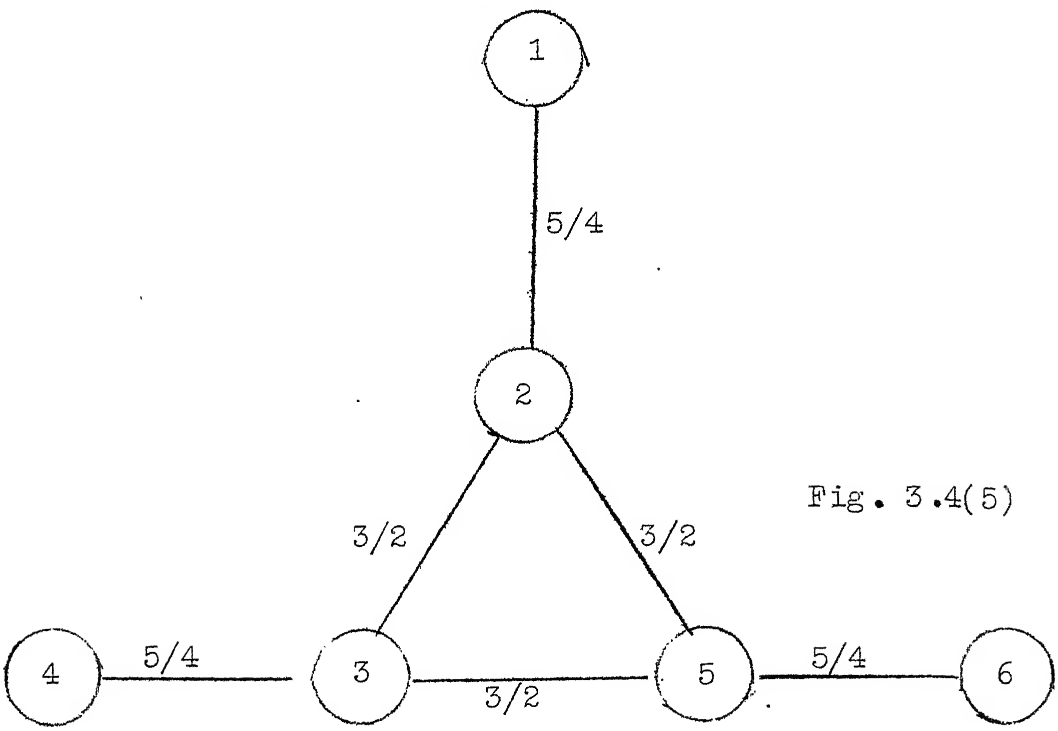


Fig. 3.4(5)

Iteration 3 :

$$\max 1(3) = 3/2$$

$$\max 2(3) = 5/4$$

$$r_3 = \frac{1}{4}$$

Optimal solution for LP $[G_3; r_3\, e]$ is

$$U(3) = \begin{bmatrix} 0 \\ \frac{1}{2^3} \\ \frac{1}{2^3} \\ 0 \\ \frac{1}{2^3} \\ 0 \end{bmatrix}$$

Thus for any large number i_o, optimal solution of i_oth iteration will be

$$U(i_o) = \begin{bmatrix} 0 \\ \frac{1}{2^{i_o}} \\ \frac{1}{2^{i_o}} \\ 0 \\ \frac{1}{2^{i_o}} \\ 0 \end{bmatrix}$$

and $r(i_o) \neq 0$.

Thus if some r_i's are not integers, the algorithm may not terminate in a finite number of iterations.

Following modification of algorithm (3.3), will ensure its termination to an optimal solution of IP $[G;r]$ in a finite number of iterations.

Algorithm (3.4) :

Here we use s_i in place of r_i in the algorithm 3.3, where s_i has been defined in section(3.2). All the other steps, will

be exactly the same, as in algorithm (3.3).

Finiteness of Algorithm (3.4) :

Initially all r_{pq}'s are given to be integer numbers. Since at every iteration 'i' some of the r_{pq}'s are getting reduced by an integer number s_i, all the requirements would be met in a finite number of iterations. Hence this algorithm (3.4) will terminate after finite number of iterations.

Let here also U(i) denote an optimal solution of the subproblem LP $[G_i; s_i e]$ of i^{th} iteration. By definition of the subproblems for i = 1,2,...,k, it is easy to see that U* is a feasible solution to LP $[G; r]$.

We will now show that property P1 (proved for algorithm (3.3)) will hold in this modified algorithm (3.4) also.

Proof : Since r_{pq}'s are given to be integers, r_i can become a fraction only if solution of some preceding subproblem was non-integer. Let i_o denote the first iteration for which $U(i_o)$ is (integer, integer/2) valued. Thus $\forall$ $j \leq i_o$, max 1(j) and max 2(j) both are integers and $s_j = r_j$, therefore P1 holds upto i_o^{th} iteration. Hence max $1(i_o+1)$ is integer and max $1(i_o+1)$ = max $2(i_o)$. If max $2(i_o+1)$ is also integer, s_{i_o+1} will again be same as r_{i_o+1} and therefore P1 will hold at $(i_o+1)^{th}$ iteration also. We now consider the case when max $2(i_o+1)$ is not integer. Here in the modified algorithm (3.4), for $(i_o+1)^{th}$ iteration. We replace r_{i_o+1} by s_{i_o+1},

where s_{i_o+1} = max $1(i_o+1)$ − max $2(i_o+1) + \frac{1}{2}$

We will now show that P1 holds for $(i_o+1)^{th}$ iteration also; i.e.

$$\max 1(i_o+2) = \max 2(i_o+1) - \frac{1}{2} ,$$

and $\max 2(i_o+1) - \frac{1}{2}$ is the requirement on all the binding edges of G_{i_o+1}. In $r(i_o+1)$, updated requirements for the binding edges of G_{i_o+1} are

$$= \max 1(i_o+1) - s_{i_o+1}$$

$$= \max 2(i_o+1) - \frac{1}{2} .$$

It is easy to see that in $r(i_o+1)$, there can not be any component having value greater than $\max 2(i_o+1)$. Therefore it will be sufficient to show that

Lemma (3.4.1) : $\max 2(i_o+1) - \frac{1}{2} = \max\limits_{(p,q)\in G} \{r_{pq}(i_o+1)\}$.

Proof : Edges of LP $[G_{i_o}; s_{i_o} e]$ which are binding w.r.t. $U(i_o)$, either form one connected subgraph of G_{i_o} or two or more components. As proved in theorem (2.3.1) either all the nodes of a component are integer valued in $U(i_o)$ or all are integer/2 valued. Let $J(i_o)$ be the index set of all the above components having noninteger valued nodes and let F_j ; $j \in J(i_o)$, denote these fractional components. From the definition of i_o, it is clear that every node p of $\bigcup\limits_{j \in J(i_o)} F_j$ is fractional valued in $U(1) + U(2) + \dots + U(i_o)$. Hence all the edges of G, having fractional requirements in $r(i_o)$ must be incident upon a node of $\bigcup\limits_{j \in J(i_o)} F_j$. Since P1 holds for i_o th iteration, G_{i_o+1} will

contain all the edges of $\bigcup_{j \in J(i_o)} F_j$ and all those other edges (p,q) of G for which $r_{pq}(i_o) = \max 1(i_o+1)$. We will now prove the following lemma :

<u>Lemma</u> (3.4.2) : No new edge of G (i.e. edge which is not in G_{i_o}) incident upon a node of $\bigcup_{j \in J(i_o)} F_j$ can be present in G_{i_o+1}.

<u>Proof</u> : Let (p_1,p_2) be an edge which was not in G_{i_o}, and is incident upon a node of $\bigcup_{j \in J(i_o)} F_j$.

Since $(p_1,p_2) \notin G_{i_o}$

$$r_{p_1p_2}(i_o-1) \leq \max 2(i_o).$$

Since $r_{p_1p_2}(i_o) = \max \{r_{p_1p_2}(i_o-1)-u_{p_1}(i_o)-u_{p_2}(i_o),\ 0\}$ at least one node from p_1 and p_2 will be in $\bigcup_{j \in J(i_o)} F_j$, therefore

$$u_{p_1}(i_o) + u_{p_2}(i_o) \geq \tfrac{1}{2}\ .$$

Hence
$$\begin{aligned} r_{p_1p_2}(i_o) &\leq \max 2(i_o) - \tfrac{1}{2} \\ &< \max 1(i_o+1) \end{aligned}$$

$\therefore\ (p_1,p_2) \notin G_{i_o+1}$.

We now continue with the proof of lemma (3.4.1). As a result of lemma (3.4.2) we conclude that all the fractional valued components F_j, $j \in J(i_o)$ of G_{i_o} are also present in

G_{i_o+1} as its components. Since for solving the subproblem $LP[G_{i_o}; s_{i_o} e]$, we first solve $LP[G_{i_o}; e]$ and then multiply the solution vector by s_{i_o}, restriction of $\frac{1}{s_{i_o+1}} U(i_o+1)$ to F_j is same as the restriction of $\frac{1}{s_{i_o}} U(i_o)$ to F_j. Therefore $\forall\ p \in \bigcup_{j \in J(i_o)} F_j$

$$u_p(i_o+1) = \frac{s_{i_o+1}}{2} \qquad 3.4(1)$$

Hence the requirement on any edge incident upon a node of $\bigcup_{j \in J(i_o)} F_j$ gets updated by atleast $s_{i_o+1}/2$ in $r(i_o+1)$.

Now let (p,q) be an edge of G

(a) $(p,q) \in G_{i_o+1}$

$$\Longrightarrow r_{pq}(i_o) = \max 1(i_o+1)$$

therefore
$$r_{pq}(i_o+1) = \max\{\max 1(i_o+1) - u_p(i_o+1) - u_q(i_o+1), 0\}$$
$$\leq \max\{\max 1(i_o+1) - s_{i_o+1}, 0\}$$
$$\leq \max 2(i_o+1) - \tfrac{1}{2}\ .$$

(b) Let $(p,q) \in G - G_{i_o+1}$.

Since $(p,q) \notin G_{i_o+1}$

$$\Longrightarrow r_{pq}(i_o) \leq \max 2(i_o+1)$$

Again two cases are possible

either (i) $r_{pq}(i_o) < \max 2(i_o+1)$

or (ii) $r_{pq}(i_o) = \max 2(i_o+1)$

Case (i) $r_{pq}(i_o) < \max 2(i_o+1)$

$$\Longrightarrow r_{pq}(i_o+1) \leq \max 2(i_o+1) - \frac{1}{2} .$$

Case (ii) $r_{pq}(i_o) = \max 2(i_o+1) =$ a fractional value.

Then by 3.4(1)

$$r_{pq}(i_o+1) \leq \max 2(i_o+1) - \frac{s_{i_o+1}}{2}$$

$$\leq \max 2(i_o+1) - \frac{1}{2}$$

Hence

$$\max 2(i_o+1) - \frac{1}{2} = \max_{(p,q) \in G} \{r_{pq}(i_o+1)\}$$

Thus $\max 1(i_o+2) = \max 2(i_o+1) - \frac{1}{2}$ and P1 is true for $(i_o+1)^{th}$ iteration.

In short we can say that P1 holds upto $(i_o+1)^{th}$ iteration because of the following property :

Property 0 : For iteration i_o all the edges having fractional requirements in $r(i_o)$ are incident upon a node of $\bigcup_{j \in J(i_o)} F_j$ and therefore their requirements get updated by at least $\frac{1}{2}$ in $U(i_o+1)$.

Now to prove that P1 holds for other iterations $j \geq i_o+1$ also, we will show that every requirement vector $r(j)$, $j \geq i_o+1$ has the property 0. Consider the $(i_o+2)^{th}$ iteration. Two cases are possible :

(a) max $2(i_0+2)$ is integer. In this case,

$$s_{i_0+2} = r_{i_0+2} = \max 1(i_0+2) - \max 2(i_0+2)$$

therefore P1 holds for $(i_0+2)^{th}$ iteration.

(b) max $2(i_0+2)$ is noninteger. Let (p_1,p_2) be an edge of G having requirement equal to max $2(i_0+2)$ in $r(i_0+1)$. Consider the following cases :

(i) s_{i_0+1} is even

(ii) s_{i_0+1} is odd.

Case (i) : In this case $U(i_0+1)$ will be integer valued. Therefore only the nodes of $\bigcup_{j \in J(i_0)} F_j$ will have noninteger values in $U(1) + U(2) + \dots + U(i_0+1)$. Hence any edge of G which has noninteger requirement in $r(i_0+1)$, is incident upon a node of $\bigcup_{j \in J(i_0+1)} F_j$, where $J(i_0+1) = J(i_0)$.

Hence property 0 holds for $(i_0+1)^{th}$ iteration. By using lemma 3.4.2 it can be proved that all the components F_j, $j \in J(i_0+1)$ will be present in G_{i_0+2} as its components. Therefore in $U(i_0+2)$, every node of $\bigcup_{j \in J(i_0+1)} F_j$ gets a value $\frac{s_{i_0+2}}{2} \geq \frac{1}{2}$ and hence

$$r_{p_1p_2}(i_0+1) \leq \max 2(i_0+1) - \tfrac{1}{2} .$$

Thus P1 holds for $(i_0+2)^{th}$ iteration also. With the help of arguments made for $(i_0+1)^{th}$ iteration, we can show that property 0 will hold for all the iterations $i > (i_0+1)$ also. Hence property P1 will continue to hold.

Case (ii) : s_{i_o+1} is odd, therefore $\frac{s_{i_o+1}}{2}$ is fraction. All the nodes of $\bigcup_{j \in J(i_o)} F_j$ will get the value $\frac{s_{i_o+1}}{2}$ in $U(i_o+1)$. Therefore all the nodes of $\bigcup_{j \in J(i_o)} F_j$ will have integer values in $U(1) + \ldots + U(i_o+1)$, and $J(i_o+1) \bigcap J(i_o)$ will be empty. Hence it can be stated that all the edges having fractional requirements in $r(i_o+1)$ are incident upon a node of $\bigcup_{j \in J(i_o)} F_j$. Thus property 0 again holds for $(i_o+1)^{th}$ iteration and therefore P1 also holds for $(i_o+2)^{th}$ iteration. We can now treat $(i_o+3)^{rd}$ iteration as the $(i_o+1)^{st}$ iteration and repeat the arguments to show that P1 holds, for every i $(i = 1,2,\ldots,k)$.

$\square$

All the other four properties P2, P3, P4 and P5 follow from P1, therefore they hold in this case also. Hence theorem (3.3.3) is also true for the case when some r_i's are fractions.

Theorem (3.3.4) :

$U^* = U(1) + \ldots + U(k)$ is an optimal solution for LP $[G;r]$.

Proof : Feasibility and optimality of U^* can be proved on the same lines as in the proof for U^* in algorithm (3.3).

$\square$

3.5

Algorithm (3.4) to solve $LP[G;\bar{r}]$

Input $G \equiv (N,E)$

requirements $\{r_{pq}\}$; $\forall (p,q) \in E$

costs c_p ; $\forall p \in N$.

Step 0 : Initialize index count : $i \leftarrow 0$

$r_{pq}(i) \leftarrow r_{pq}$

$E(i) \leftarrow E$

Step 1 : $i \leftarrow i+1$

$$\max 1(i) \leftarrow \max_{(p,q) \in E} \{r_{pq}(i-1)\}$$

$$E(i) \leftarrow \{(p,q) \mid r_{pq}(i-1) = \max 1(i)\}$$

$$\max 2(i) \leftarrow \max_{(p,q) \in E-E(i)} \{r_{pq}(i-1)\}$$

If max 2(i) is zero, go to step 6.

If max 2(i) is integer, go to step 3.

Otherwise go to step 2.

Step 2 : $s_i \leftarrow \max 1(i) - \max 2(i) + \frac{1}{2}$

go to step 4 .

Step 3 : $s_i \leftarrow \max 1(i) - \max 2(i)$

Step 4 : Let $G_i \equiv (N, E(i))$

to solve $LP[G_i;e]$:

(4.1) Solve $DLP[\bar{G}(G_i);e]$

(4.2) Construct an optimal solution for $LP[\bar{G}(G_i);e]$

(4.3) Construct an optimal solution $U(i)$ for

$LP[G_i; s_i\, e]$ where

$$U(i) = \frac{U^1 + U^2}{2} \qquad s_i \,.$$

Step 5 :

$$r_{pq}(i) \leftarrow \max \begin{cases} r_{pq} - u_p(i) - u_q(i) \\ 0 \end{cases}$$

go to step 1.

Step 6 : $u_p^* = \sum_{j=1}^{i} u_p(j)$

stop.

CHAPTER IV

SOME PROBLEMS RELATED TO P[G;r]

4.1 INTRODUCTION

In the present chapter we identify some problems such as transportation problem, generalized vertex packing problem etc., which can also be solved using the decomposition techniques discussed in previous chapters.

In section 4.2, we develop an algorithm for P[G;r], when G is a tree. In this integer optimal solutions for decomposed subproblems of LP[G;r] are obtained by using dynamic programming. An attempt is also made to calculate an upper bound on the number of subproblems to be solved.

In section 4.3, a natural modification of the algorithm (3.4), when the underlying graph G is bipartite, is presented. For an example dual of the standard transportation problem is of the form LP[G;r] on a bipartite graph G.

In section 4.4, we again deal with a generalized edge covering problem (BP), when upper bounding constraints on its variables have also been added to it. It is shown that BP can also be solved, by solving its linear relaxation and a weighted edge covering problem on a subgraph of G. Moreover linear relaxation of BP can again be solved by the algorithm (3.4) after making some minor modifications.

In section 4.5, we discuss the generalized vertex packing problem (GVP). Here it is shown that by a simple linear transformation, this can be transformed to a problem of the form BP and thus can be solved by the algorithm modified for BP.

In section 4.6, we take a problem whose constraints are mixture of constraints of generalized vertex packing and generalized edge covering problems. We show that this problem can also be transformed to a BP.

4.2 ALGORITHM FOR $P[G;r]$, WHEN G IS A TREE

Here an integer valued optimal solution of $LP[G;r]$ is obtained by solving its subproblems which are generated by using the same decomposition technique as described in chapter 3. For solving each such subproblem, a dynamic programming type of algorithm is suggested which guarantees to give an integer valued optimal solution. Thus by solving $LP[G_i;s_ie]$ finite number of times, an integer valued optimal solution of $LP[G;r]$ can be obtained and thus there is no need of solving a weighted edge covering problem.

Since G is a tree, underlying graph of each subproblem will also be a loopless graph. In case, G_i is disconnected, $LP[G_i;s_ie]$ will get reduced to independent LP's on these components and therefore can be solved seperately. Hence without loss of generality, we assume that G_i is connected. Solution U obtained by the following algorithm will be shown to be optimal and hence $U(i) = U$.

4.2.1 Notations and Definitions

Outer node	:	node having degree one in G
S_1	:	A subset of the node set N of G_i
$T_i(p)$	:	A maximal connected subgraph of G_i containing only the node p and nodes from the set S_1.
deg(p)	:	Degree of p in T_i, where T_i is a subgraph of G_i.
N(p)	:	$\{q \mid (p,q) \in G_i\}$
$\bar{N}(p)$	:	$\{q \mid q \in S_1 \text{ and } (p,q) \in G_i\}$
c_p^1	:	Value of the objective function for the optimal solution of $LP[T_i(p); e]$ provided $u_p=1$, where e is a summation vector corresponding to the edge set of $T_i(p)$. $c_p^1 = c_p + \sum_{q \in \bar{N}(p)} \bar{c}_q$
c_p^o	:	Value of the objective function for the optimal solution of $LP[T_i(p); e]$ provided $u_p = 0$. Here $c_p^o = \sum_{q \in \bar{N}(p)} c_q^1$
$\bar{c}_p$		$\min\{c_p^1, c_p^o\}$

4.2.2 Algorithm (4.2.2) Input $[G_i; c_p, \forall p \in G_i]$

Step 0 $\quad S_1 \leftarrow \phi$

$\quad T_i \leftarrow G_i$

Step 1 $\quad S_o \leftarrow \{p \mid p \in T_i;\ \deg(p) = 1\}$

Find $\bar{N}(p)$ for every node p of S_o

$\bar{N}(p) \leftarrow \{q \mid q \in S_1 \text{ and } (p,q) \in G_i\}$

$c_p^1 \leftarrow c_p + \sum_{q \in \bar{N}(p)} \bar{c}_q \quad \forall\ p \in S_o$

$c_p^o \leftarrow \sum_{q \in \bar{N}(p)} c_q^1 \ \forall\ p \in S_o$

$\bar{c}_p \leftarrow \min\{c_p^1, c_p^o\}$

$S_1 \leftarrow S_1 \cup S_o$

$T_i \leftarrow T_i/S_o$, where T_i/S_o is the graph T_i, left after removing all the nodes of S_o and edges incident on them.

<u>Step 2</u> (i) If T_i contains exactly one node go to step 4.

(ii) If T_i contains exactly two nodes, go to step 3.

(iii) Otherwise go to step 1.

<u>Step 3</u> Calculate c_p^1, c_p^o for every $p \in T_i$ as follows

$$c_p^1 = c_p + \sum_{q \in \bar{N}(p)} \bar{c}_q$$

$$c_p^o = \sum_{q \in \bar{N}(p)} c_q^1$$

Let the nodes of T_i be p_o and q_o.

Find $W_{L_1L_2} = c_{p_o}^{L_1} + c_{q_o}^{L_2}$ where $L_1+L_2 \geq 1$ and $L_1, L_2 \in \{0,1\}$.

Let $W_{J_1J_2} = \min_{L_1+L_2 \geq 1} \{W_{L_1L_2}\}$

Set $u_{p_o} = J_1$

$u_{q_o} = J_2$

and go to step 5.

<u>Step 4</u> Let p_o be the only node of T_i.

Find $c^1_{p_o}$, $c^o_{p_o}$ and $\bar{c}_{p_o}$.

If $\bar{c}_{p_o} = c^J_{p_o}$ where $J \in \{0,1\}$.

Set $u_{p_o} = J$ and go to step 5.

<u>Step 5</u> We now give values to all the other nodes of G_i.

Select a node p which has been given a value.

(i) If $u_p = 0$, set $u_q = 1$ $\forall$ node $q \in N(p)$.

(ii) If $u_p = 1$ give values to all other nodes of $\bar{N}(p)$ which have not been given values till now, as follows:

If $\bar{c}_q = c^1_q$ where $q \in \bar{N}(p)$

Set $u_q = 1$

Otherwise, set $u_q = 0$.

If all the nodes of G_i have been given values, stop; otherwise again go to step 5.

4.2.3 <u>Proof of optimality</u>

It will be sufficient to give only an outline of the proof for optimality of the solution obtained by the algorithm (4.2.2).

From the description of the algorithm itself, it is easy to see that for any node p, $T_i(p)$ at the time when c^1_p, c^o_p, $\bar{c}_p$ are calculated, is uniquely defined in such a way that if $q \neq p$, then exactly one of the following is true:

(i) $T_i(q) \subset T_i(p)$

(ii) $T_i(p) \subset T_i(q)$

(iii) $T_i(q) \cap T_i(p) = \phi$.

From the definition of c_p^o, c_p^1 and $\bar{c}_p$, it is obvious that $\bar{c}_p$ is the optimal value of LP $[T_i(p); e]$. Now consider the stage, when the number of nodes in $T_i(p)$ is either one or two.

Case 1 Let T_i contain only one node, say it is node p_o. Obviously $T_i(p_o) = G_i$ and hence $\bar{c}_{p_o}$ is the optimal value of LP $[G_i; e]$.

Case 2 Let T_i contain only two nodes, p_o and q_o. In this case, $T_i(p_o) \bigcup T_i(q_o) = G_i$.
We then give values to the nodes p_o and q_o, such that $u_{p_o} + u_{q_o} \geq 1$ and the total cost for $T_i(p_o) \bigcup T_i(q_o)$ is $\min \left\{ c^1_{q_o} + c^o_{p_o},\ c^1_{q_o} + c^1_{p_o},\ c^o_{q_o} + c^1_{p_o} \right\}$.

4.2.4 Number of computations required to solve a subproblem LP $[G_i; e]$

For each node q of the graph of a subproblem, we have to calculate c_q^1, c_q^o, $\bar{c}_q$. For this, only additions and comparisions are needed. In particular
deg(q) additions are necessary for computing c_q^1
deg(q)-1 additions are necessary for computing c_q^o
One comparision is necessary for computing $\bar{c}_q$.
Hence the total number of operations to compute c_q^1, c_q^o, $\bar{c}_q$ for $q = 1,\ldots,k$, where k is cardinality of the node set of G_i

$$\leq 2 \sum_{q=1}^{k} \deg(q) - k + k$$

$$\leq 4(k-1) \qquad \text{(since G is a tree)}$$

$$\approx O(k).$$

e_i may be dropped out because of being nonbinding in a subsequent problem and it may again reappear in a later problem. Let N_i denote the number of subproblems P^r of A in which e_i reappears for the first time, i.e., $e_i \in P^r$ and $e_i \notin P^{r-1}$ for some r such that $I_o+1 < r \leq I_o+ I(k)$; or for $r = I_o+1$ it appears in the problem P^{I_o+1}. Since each subproblem belonging to A, differs from its immediate predecessor, there is atleast one edge in each, which reappears in it for the first time, in the above sense.

Hence $$I(k) \leq \sum_{e_i \in G^k} N_i$$

Computation of N_i

We will first find N_i for the case when, G^k is connected. It will be easy to see from this derivation that the same value of N_i can also be taken as an upperbound for N_i, for the case, when G^k is a forest.

Let $e_o = (p_o, q_o)$ be an edge of G^k. Deletion of the edge e_o from G^k partitions it into two components $S_1(e_o)$ and $S_2(e_o)$. Let $p_o \in S_1(e_o)$ and $q_o \in S_2(e_o)$. Now we indicate some observations from the algorithm which will be used for the derivation of N_i.

Here the subproblems referred to, have the subgraphs of G^k as their underlying graphs.

(1) An edge $e_o = (p_o, q_o)$ gets dropped from P^{s_o}, where $I_o+ 1 \leq s_o \leq I_o + I(k)$ if,

$$r_{p_o q_o}(s_o) < \max 1\ (s_o+1) \qquad 4.2.5(a)$$

where $r_{p_o q_o}(s_o)$ is the requirement on the edge (p_o, q_o), updated after s_o^{th} iteration.

Inequality 4.2.5(a) is possible only if

$$u_{p_o}(s_o) > 0$$

and $$u_{q_o}(s_o) > 0$$

(2) e_o reappears in P^{t_o} after being dropped out from P^{s_o} only if $r_{p_o q_o}(t_o-1) = \max 1\ (t_o)$

Since at any iteration s, $(s_o < s < t_o)$

$$r_{p_o q_o}(s-1) = r_{p_o q_o}(s-2) - u_{p_o}(s-1) - u_{q_o}(s-1)$$

hence $r_{p_o q_o}(s-1) < \max 1\ (\ s\ )$.

This is so because 't_o' is the first iteration after s_o^{th} iteration where e_o reappears. Moreover if either u_{p_o} or u_{q_o} is positive in $U(s-1)$, then

$$u_{p_o}(s-1) + u_{q_o}(s-1) \geq \max 1\ (s-1) - \max 2\ (s-1)$$

Here in every iteration j, $\max 1(j) - \max 2(j)$ is always an integer and $\max 1\ (i+1) = \max 2(i)$. Therefore edge e_o can not reappear unless somewhere between P^{s_o} and P^{t_o} $u_{p_o}(s) = u_{q_o}(s) = 0$ for some s.

(3) Let $C1(s_o, e_o)$ and $C2(s_o, e_o)$ be two components into which the underlying graph of P^{s_o} gets divided after s_o^{th}

iteration. Let the node p_o be in $C1(s_o,e_o)$ and let q_o be in $C2(s_o,e_o)$.

As long as the components $C1(s_o,e_o)$ and $C2(s_o,e_o)$ continue to be the components of the subproblems that follow P^{s_o}, the nodes p_o and q_o will get positive values because optimal solution for the LP on this component is obtained with the help of previous problem, and hence the edge e_o will not reappear.

Thus if e_o reappears in t_o^{th} iteration, by observation 2 at least one edge from $S_1(e_o) - C1(s_o,e_o)$ and one from $S_2(e_o) - C2(s_o,e_o)$ should be added to $C1(s_o,e_o)$ and $C2(s_o,e_o)$ respectively to change the structure of these components.

(4) Any edge $e_j = (p_j,q_j)$ of G^k, such that either p_j or q_j is an outer node in G^k, will be present in every problem of A. This is so because $c_j > 0 \ \forall j \in G^k$.

Now using the above properties, we will prove the following theorem:

<u>Theorem (4.2.5)</u> $N_i \leq 1 + \min \{|S_1(e_i)|, |S_2(e_i)|\}$

where $|S_j(e_i)|$ = number of edges in $S_j(e_i)$ for $j = 1,2$.

<u>Out line of the proof</u>

Let $P^{i_1}, P^{i_2},\ldots,P^{i_{N_i}}$ be all the subproblems of A, where e_i reappears for the first time. With each of the P^{i_s}, $s = 2,\ldots,N_i$ we will be able to associate two edges f_s and g_s such that,

(1) (a) $f_s \in S_1(e_i)$ · (b) $g_s \in S_2(e_i)$

where $S_1(e_i)$, $S_2(e_i)$ are the components of G^k, obtained by deleting e_i from it.

and

(2) $f_s \neq f_j \quad j = 2,3,\ldots,s-1$

$g_s \neq g_j \quad j = 2,3,\ldots,s-1$

We will therefore be able to conclude that:

$$N_i \leq 1 + \min \{|S_1(e_i)|, |S_2(e_i)|\} .$$

Proof: For the sake of easiness, let us enclose some of the problems of A into boxes as follows:

(i) In the first problem of each box, edge e_i reappears for the first time and it is also present in each problem contained in a box.

(ii) In the last problem of a box, edge e_i becomes non binding and therefore gets dropped.

(iii) Any problem of A, which is not contained in any box, does not contain the edge e_i.

Let P^{i_r} and P^{j_r} denote the first and the last problem, respectively of the r^{th} box.

We will first show that with each P^{i_s}; $s = 2,3,\ldots,N_i$, we can associate an edge f_s such that $f_s \in S_1(e_i)$ and it must have been introduced in at least one problem of A, say in P^t, before the problem P^{i_s}, and it has played a role in

changing the value of node p_i through a series of changes between problems $P^{j_{s-1}}$ and P^{i_s} only. We can call such a change as cumulative chain effect of f_s on p_i. It will be denoted as $CHE(f_s, t)$ where t is the iteration where f_s was introduced and $j_{s-1} < t < i_s$.

This type of selection of edges will guarantee that f_s has not been assigned to any problem of A before P^{i_s}.

g_s's can also be assigned in the same fashion.

Note: Reappearance of an edge e_i occurs due to the change of the values of p_i and q_i both. f_s and g_s are the first edges which can be considered to be responsible for changing the structure of components containing p_i and q_i respectively, after e_i was dropped out. Thus f_s and g_s can be considered to be responsible for the change in values from 1 to 0 of p_i and q_i, before e_i reappears in P^{i_s}.

Assignment of f_s and g_s to P^{i_s}, $(s = 2, \ldots, N_i)$

We will first assign f_s to P^{i_s} and then in the same manner g_s can also be assigned to P^{i_s}. Consider any two boxes s_b and s_b+1. Then edge e_i becomes nonbinding in $P^{j_{s_b}}$, with respect to its optimal solution. Consider the components formed by the binding edges of $P^{j_{s_b}}$. Let $C1(j_{s_b}, e_i)$ and $C2(j_{s_b}, e_i)$ be two such components containing p_i and q_i respectively, where $e_i=(p_i, q_i)$. Since e_i reappears in $P^{i_{s_b+1}}$, it implies that between

$P^{j_{s_b}}$ and $P^{i_{s_b+1}}$, at least one edge from $S_1(e_i) - C1(j_{s_b}, e_i)$ has been added to $C1(j_{s_b}, e_i)$. Let $e_1 = (m_1, n_1)$ be one such edge which has first changed the structure of $C1(j_{s_b}, e_i)$. Let $n_1 \in C1(j_{s_b}, e_i)$, and suppose this edge e_1 was introduced in P^{t_1}, where $j_{s_b} < t_1 < i_{s_b+1}$.

Since $n_1 \in S_1(e_i)$

$$S_1(e_1) < S_1(e_i).$$

There are now two possible cases:

Case a: e_1 is a new edge, i.e., in any problem of A, before P^{t_1}, e_1 has not been introduced. In this case take $j = 1$ and go to step b.

Case b :

Step a e_1 is an old edge. By an old edge, we mean that in at least one problem of A, before P^{t_1} it has also been dropped out. Let s_1 be the largest number, but less than t_1, such that in the optimal solution of P^{s_1}, e_1 has become nonbinding. Obviously $s_1 < j_{s_b}$, because the underlying graph of every subproblem between j_{s_b} and t_1^{th} iterations, contain the component $C1(j_{s_b}, e_i)$ as a component and e_1, which is not contained in $C1(j_{s_b}, e_i)$, but is incident upon a node of $C1(j_{s_b}, e_i)$. Let $C1(s_1, e_1)$ and $C2(s_1, e_1)$ be two components containing m_1 and n_1 respectively and are formed from the edges of P^{s_1} which are binding with respect to its optimal solutions. Since e_1 reappears for the first time in P^{t_1},

it implies that between P^{s_1} and P^{t_1}, at least one edge from $S_1(e_1) - Cl(s_1,e_1)$ which is incident upon a node of $Cl(s_1,e_1)$ has been added to $Cl(s_1,e_1)$.

Let $e_2 = (m_2,n_2)$, where $n_2 \in Cl(s_1,e_1)$ be one such edge which has first changed the structure of $Cl(s_1,e_1)$.

Let it appear for the first time in t_2^{th} iteration, where $s_1 < t_2 < t_1$.

Since $n_2 \in S_1(e_1)$

$$S_2(e_2) \subset S_1(e_1) \subset S_1(e_i).$$

For e_2 again there may be two cases:

Case b_1 e_2 is a new edge.

Case b_2 e_2 is an old edge.

Case b_1 e_2 is a new edge, therefore set $j = 2$ and go to step b.

Case b_2 Let s_2 be the largest number less than t_2, such that in P^{s_2}, e_2 becomes nonbinding. Now we repeat the procedure suggested above for an old edge e_1.

Say, after repeating this procedure j times, we get a sequence of following edges

$$e_1, e_2, \ldots, e_j$$

Note: For any r; $1 < r \leq j$, $e_r = (m_r,n_r)$ was introduced in P^{t_r}, such that e_{r-1} had already been dropped out, i.e., $s_{r-1} < t_r < t_{r-1}$ and $n_r \in Cl(s_{r-1},e_{r-1})$. By the choice of

these edges, it is easy to see that

$$S_1(e_j) \subset S_1(e_{j-1}) \subset \ldots \quad S_1(e_i) \qquad 4.2.5(b)$$

We now claim that the above repetition of step a, will be done only a finite number of times. Proof is as follows: Since $|S_1(e_i)|$ is an integer, from 4.2.5(b) it is easy to see that we will finally get an edge e_j which reappears for the first time in P^{t_j} and for which $|S_1(e_j)| = \phi$ and $n_j \in C1(s_{j-1}, e_{j-1})$. Hence m_j is an outer node in G^k. By above observation 4, e_j reappears for the first time in the first problem of A and will be present in every problem of A. Therefore e_j is a new edge for $P^{t_j} = P^{I_o+1}$. Thus finally we will get a new edge e_j, such that step a is not be repeated any further.

Now go to step b.

Step b We will first prove that the cumulative chain effect of the introduction of e_j in P^{t_j} can reach the node p_i only in between the problems $P^{j_{s_b}}$ and $P^{i_{s_b+1}}$.
In P^{t_j}, e_j has been introduced for the very first time i.e., before P^{t_j}, no problem of A contains e_j.

(i) If $j = 1$

$$e_1 = (m_1, n_1) \quad \text{where} \quad n_1 \in C1(j_{s_b}, e_i).$$

It is obvious that CHE (e_j, t_j) can reach the node p_i belonging to $C1(j_{s_b}, e_i)$ only in between $P^{j_{s_b}}$ and $P^{i_{s_b+1}}$. Therefore, we can take $f_{s_b} = e_j$.

(ii) If $j > 1$

(a) Consider the interval $[P^{t_j}, P^{t_{j-1}})$, which contains the problems P^i such that $t_j \leq i < t_{j-1}$.

Since e_j is introduced in P^{t_j} for the very first time, and $n_j \in C1(s_{j-1}, e_{j-1})$

$$CHE\ (e_j, t^j) \rightarrow C1(s_{j-1}, e_{j-1}) \qquad 4.2.5(c)$$

i.e., CHE (e_j, t^j) can effect a node of $C1(s_{j-1}, e_{j-1})$.

Now $e_j \in S_1(e_{j-1})$ and e_{j-1} which is a connecting edge does not belong to any problem of $[P^{t_j}, P^{t_{j-1}})$.

Therefore CHE $(e_j, t^j) \not\rightarrow S_2\ (e_{j-1})$

i.e., CHE (e_j, t^j) can not change the value of any node of $S_2(e_{j-1})$.

Since node $p_i \in S_2(e_{j-1})$,

$$CHE\ (e_j, t^j) \not\rightarrow p_i \text{ in } [P^{t_j}, P^{t_{j-1}}) \qquad 4.2.5(d)$$

(b) Now consider the interval $[P^{t_{j-1}}, P^{t_{j-2}})$.

Since $n_{j-1} \in C1(s_{j-2}, e_{j-2})$

CHE $(e_{j-1}, t^{j-1}) \rightarrow C1(s_{j-2}, e_{j-2})$

Since m_{j-1} is present in $C1(s_{j-1}, e_{j-1})$ by 4.2.5(c), we can say,

$$CHE(e_j, t^j) \rightarrow C1(s_{j-1}, e_{j-1}) \cup C1(s_{j-1}, e_{j-2})$$

e_{j-2} does not belong to any problem of $[P^{t_{j-1}}, P^{t_{j-2}})$, and e_j and e_{j-1} both are in $S_1(e_{j-2})$, therefore

$CHE\ (e_{j-1}, t^{j-1}) \not\to S_2(e_{j-2})$

Now $S_2(e_{j-2}) \subset S_2(e_{j-1})$

Therefore $CHE\ (e_j, t^j) \not\to S_2(e_{j-2})$.

Since p_i is also in $S_2(e_{j-2})$, therefore

$$CHE\ (e_j, t^j) \not\to p_i \qquad 4.2.5(e)$$

Here we can say that $CHE\ (e_j, t^j)$, can not reach the node p_i before $P^{t_{j-2}}$.

(c) Proceeding this way till the interval $[P^{t_2}, P^{t_1})$, we can conclude that

$$CHE\ (e_j, t^j) \to C1(s_{j-1}, e_{j-1}) \cup \ldots \cup C1(s_1, e_1) \qquad 4.2.5(f)$$

and $CHE\ (e_j, t^j) \not\to p_i$ before P^{t_1} $\qquad$ 4.2.5(g)

(d) Finally consider the interval $[P^{t_1}, P^{i_{s_b+1}})$

Since $n_1 \in C1(j_{s_b}, e_i)$

and $m_1 \in C1(s_1, e_1)$

By 4.2.5(f), we can say that

$$CHE\ (e_j, t^j) \to C1(j_{s_b}, e_i)$$

Since p_i is also in $C1(j_{s_b}, e_i)$

$$CHE\ (e_j, t^j) \to p_i.$$

Hence by 4.2.5(g) we conclude that the cumulative chain effect of the introduction of e_j in P^{t_j} can reach the node of p_i only in between $P^{j_{s_b}}$, $P^{i_{s_b+1}}$. Hence we take

$$f_{s_b+1} = e_j$$

Similarly g_s's can also be assigned to each first problem of a box.

Hence we can say,

$$N_i \leq 1 + \min \{|S_1(e_i)|, |S_2(e_i)|\} \qquad \square$$

<u>Theorem 4.2.5</u> $L \approx O(n^3)$ where n is the number of nodes in G.

<u>Proof</u>: It is easy to see that the sum $\sum_{e_i \in G^k} \min \{|S_1(e_i)|, |S_2(e_i)|\}$ will be largest when e_i's form a graph having only two outer nodes. In this case, above **sum** will be:

$\leq 2\left[1+2+3+ \ldots \frac{k-2}{2}\right]$ where k is the number of nodes in G^k.

Hence

$$N_1+N_2+\ldots+N_{k-1} \leq 2\left[1+2+3\ldots \frac{k-2}{2}\right] + (k-1)$$

Therefore,

$$I(k) \leq \frac{2\left(\frac{k-2}{2}\right)\left(\frac{k-2}{2}+1\right)}{2} + (k-1)$$

$$= \frac{k}{4}(k-2) + (k-1)$$

$$\text{Since} \quad L \leq I(1) + I(2) + \ldots I(n)$$

$$\leq \sum_{k=1}^{n} \frac{k}{4}(k-2) + \sum_{k=1}^{n}(k-1)$$

$$= \frac{1}{4}\left[\sum_{k=1}^{n} k^2 + 2\sum_{k=1}^{n} k - 4n\right]$$

$$= \frac{1}{4}\left[\frac{n(n+1)(2n+1)}{6} + \frac{2n(n+1)}{2} - 4n\right]$$

$$\approx O(n^3) \qquad \square$$

4.3 ALGORITHM FOR $P[G;\bar{r}]$ ON A BIPARTITE GRAPH

Consider the problem $P[G;\bar{r}]$, when G is a bipartite graph. Here a natural modification of the algorithm (3.4) is made, which gives an integer optimal solution for $LP[G;\bar{r}]$. In this case subproblems are generated by the same decomposition method as described in chapter 3.

In the algorithm (3.4) at the i^{th} iteration, to get an optimal solution of $LP[G_i;\bar{e}]$, we solve $DLP[\bar{G}(G_i);\bar{e}]$, which is a maximum flow problem on a bipartite graph generated from G_i. Here in this case, G_i itself is a bipartite graph, therefore $DLP[G_i;\bar{e}]$ is a maximum flow problem on G_i. Thus the only modification of the algorithm is that, now we solve $DLP[G_i;\bar{e}]$ instead of $DLP[\bar{G}(G_i);\bar{e}]$. From this solution, as shown in Appendix A, a (0,1) optimal solution for $LP[G_i;\bar{e}]$ is generated from which optimal integer solution of $LP[G_i, s_i\bar{e}]$ is found. Finally by adding these optimal solutions for subproblems, an integer valued optimal solution of $LP[G;\bar{r}]$ and hence an optimal solution for $P[G;\bar{r}]$ is obtained.

4.4 BOUNDED VARIABLE PROBLEM

In the present section we deal with a bounded variable version of $P[G;\bar{r}]$, which is formulated as follows:

$BP[G;\bar{r}]$

$$\text{minimize} \quad \sum_{i=1}^{n} c_i u_i$$

$$\text{such that} \quad u_i + u_j \geq r_{ij} \quad \forall\,(i,j) \in G$$

$$\left.\begin{array}{l} u_i \leq W_i \\ u_i \geq 0,\ \text{Integer} \end{array}\right. \quad \forall\, i \in G$$

where,

w_i's are nonnegative integers.

We will first show that an optimal solution of BP $[\bar{G};\bar{r}]$ can also be decomposed in two parts as we have shown for an optimal solution of $P[G;\bar{r}]$ in chapter 2.

Above result can easily be proved by proceeding on the same lines as in chapter 2 and using the following:

Since w_i's are integers, 1/2 e will be an optimal linear solution for LBP $[\bar{G}_j;\bar{e}]$ where $\bar{G}_j$ is a fractional component of $\bar{G}$ as defined in chapter 2 for LP $[\bar{G};\bar{r}]$; and LBP $[\bar{G}_i;\bar{e}]$ denotes the linear relaxation of BP $[\bar{G}_i;\bar{e}]$.

Algorithm (4.4) (To solve LBP $[\bar{G};\bar{r}]$)

As given in the algorithm (3.4), at the beginning of i^{th} iteration, we will first compute s_i. Let $\{i_1,i_2,\ldots,i_p\}$ be the set of nodes of the graph G_i of LP $[\bar{G}_i;s_i\bar{e}]$. We now find $\bar{s}_i$ as follows:

$$\bar{s}_i = \min\{w_{i_1}, w_{i_2},\ldots,w_{i_p}, s_i\}$$

At the i^{th} iteration, we then solve LBP $[\bar{G}_i;\bar{s}_i\bar{e}]$. After finding an optimal solution $U(i)$ of LBP $[\bar{G}_i;\bar{s}_i\bar{e}]$, we update r_{pq} and w_p both for all the nodes p and q of G_i. If for any node p, updated w_p becomes zero, we replace c_p by a large integer number M, so that in all future iterations u_p is forced to take zero value. We then give values to all the neighbouring nodes of p according to the updated requirements on their

adjoining arcs. Again after updating w_p and r_{pq} $\forall$ p,q of G, we proceed for the next iteration as before.

Theorem (4.4.1) Modified algorithm 4.4 gives an optimal solution for LBP $[G;r]$.

Proof. Proof is exactly on the same lines as we have given for LP $[G;r]$ in chapter 3. It is easy to see that all the five properties $P_1, P_2, \ldots, P_5$ on which whole proof is based, hold for this case also.

4.5 GENERALIZED VERTEX PACKING PROBLEM

Here we discuss the generalized vertex paking problem which can be considered to be a generalization of the vertex packing problem in the same way as the generalized edge covering problem for the edge covering problem.

Generalized vertex packing problem is formulated as follows:

GVP $[G;r]$

$$\text{maximize} \quad \sum_{p=1}^{n} c_p u_p$$

$$\text{such that} \quad u_p + u_q \leq r_{pq} \quad \forall\ (p,q) \in G$$

$$u_p \geq 0, \ \text{Integer} \quad \forall\ p \in G$$

We now show that GVP $[G;r]$ can be transformed to a BP $[G;r']$ for some r'.

Let $\bar{r} = \max\limits_{(p,q)\in G} \{r_{pq}\}$

Using the transformation $U = \bar{r}e - U'$, GVP $[G;\bar{r}]$ transforms to

$$\text{maximize} \quad \Sigma\, c_p(\bar{r} - u'_p)$$

$$\text{such that} \quad \bar{r} - u'_p + \bar{r} - u'_q \leq r_{pq} \quad \forall\ (p,q) \in G$$

$$u'_p \leq \bar{r} \quad \forall\ p \in G$$

Since $\sum\limits_{p \in G} c_p \bar{r}$ is a constant, from the selection of $\bar{r}$, it is easy to see that $u'_p \geq 0 \ \forall\ p \in G$. GVP $[G;\bar{r}]$ can also be written as:

$$\text{minimize} \quad \sum_p c_p u'_p$$

$$\text{such that} \quad u'_p + u'_q \geq 2\bar{r} - r_{pq} \quad \forall (p,q) \in G$$

$$0 \leq u'_p \leq \bar{r} \quad \forall\ p \in G$$

which is same as the problem BP $[G;\bar{\bar{r}}]$ where

$$\bar{\bar{r}}_{pq} = 2\bar{r} - r_{pq} \quad \forall (p,q) \in G.$$

Hence GVP $[G;\bar{r}]$ can also be solved by the algorithm 4.4.

4.6 MIXED CONSTRAINTS PROBLEM

Consider the problem,

GCVP $[G;\bar{r}]$

$$\text{minimize} \quad \sum_{p \in G} c_p u_p$$

$$\text{such that} \quad \left.u_p + u_q \geq r_{pq}\right\} S_1$$

$$\left.u_p + u_q \leq r_{pq}\right\} S_2$$

$$u_s \geq 0, \text{ Integer} \quad \forall\ p \in G$$

where S_1 and S_2 are subsets of the edge set E of G.

We show that this can also be transformed to a BP $[\bar{G}';r']$.

Let $\bar{r} = \max \{ r_{pq} | (p,q) \in S_2 \}$

Make a linear transformation $U = \bar{r}e - U'$ on the constraint set of S_2. GCVP $[\bar{G};r]$ is then equivalent to

$\bar{P}$.

$$\begin{array}{lll}
\text{Minimize} & \sum_p c_p u_p & \\
\text{such that} & \left.\begin{array}{l} u_p + u_q \geq r_{pq} \\ u'_p + u'_q \geq r'_{pq} \end{array}\right\} \begin{array}{l} S_1 \\ S_2 \end{array} & \\
& u'_p \leq \bar{r} & \\
& u_p \geq 0 & \forall\, p \in G \\
& u_p + u'_p = \bar{r} & \forall\, p \in G \text{ such that } (p,q) \in S_2 \text{ for some } q.
\end{array}$$

As in section 4.5, here also we can add nonnegativity restrictions on u'_p in $\bar{P}$. In order to convert the equality constraints into inequalities, we attach a large penality cost when ever the equality constraint is violated.

Thus GCVP $[\bar{G};r]$ is also equivalent to the following:

$$\begin{array}{lll}
\text{minimize} & \sum_{p \in G} c_p u_p + M \left[\sum_p (u_p + u'_p) - \bar{r} \right] & \\
\text{such that} & \left.\begin{array}{l} u_p + u_q \geq r_{pq} \end{array}\right\} S_1 & \\
& \left.\begin{array}{l} u_p + u_q \geq r'_{pq} \end{array}\right\} S_2 & \\
& u_p + u'_p \geq \bar{r} & \\
& 0 \leq u'_p \leq \bar{r} & \forall\, p \in G \\
& u_p \geq 0 &
\end{array}$$

which is of the form BP $[\bar{G}';r']$ for some graph G' and a requirement vector r'. Therefore GCVP $[\bar{G};r]$ can also be solved by the algorithm (4.4).

CHAPTER V

COMPUTATIONAL EXPERIENCES

5.1 INTRODUCTION

In this chapter we discuss computational experience with the algorithm (3.4) described in chapter three for solving linear relaxation of the generalized edge covering problem (P). Some observations on the behaviour of the algorithm (3.4) are made from the computational experience gained by solving several randomly generated test problems.

We have shown in chapter three that in the worst possible case, number of linear programming subproblems generated are bounded by the highest requirement among all the edges of G ($\max R = \max\{r_{ij}\}(i,j)\epsilon G$). Further each iteration will require at most $O(n^3)$ computations where n is the number of nodes in the underlying graph of the original problem P. Thus in the worst case the computations are bounded by $T(n) \approx \max R.O(n^3)$. In general all these generated subproblems are not needed to be solved seperately, because they have the property that if for a subproblem, optimal solution of its immediate predecessor is feasible for it, then it will be optimal also. Thus the above bound $T(n)$ would very seldom be attained. Computational results have also shown that the average number of iterations required are usually of the $O(n)$. Here by number of iterations we mean number of different subproblems solved.

5.2 COMPUTER PROGRAM

Computer program for the algorithm (3.4) is coded in FORTRAN-10-DEC-10 version and is run in DEC1020 KL 40 process system. As the main aim of the computations is to get an insight into the algorithm, the program is kept simple and is not efficient from the point of view of programming efficiency. As the program does not use the best data structure and other computer techniques, the computational time is not considered as an important index of performance. This program is structured with the help of following three main subroutines:

a Subroutine 'GEN'

b Subroutine 'ITER'

c Subroutine 'VALUES'

a Subroutine 'GEN' It generates random test LP-problems for the given values of n, de and max R, where

n = number of nodes in G

and de = density of the graph G i.e.

$$= \frac{\text{number of edges present in G}}{\text{possible number of edges on n nodes}}$$

Random graphs G are generated with the help of pseudo random numbers which are distributed uniformly over $[0,1]$.

b Subroutine 'ITER' This subroutine generates subproblems for all the iterations. A subproblem of i^{th} iteration is generated by computing max1(i), max2(i) and s_i as described in

chapter three for the LP problem having requirements updated after $(i-1)^{th}$ iteration.

c Subroutine 'VALUES' It is used for solving a subproblem generated by the subroutine 'ITER'. It first generates a symmetric bipartite graph corresponding to the underlying graph of this subproblem and then it solves the maximum flow problem on this bipartite graph by Ford and Fulkerson's labelling method [18]. There are other efficient algorithms available to solve the maximum flow problem such as, Dinic [13], Malhotra [38], but for the sake of program simplicity Ford and Fulkerson's method has been selected.

From the optimal solution of this maximum flow problem, a (0,1/2,1) valued optimal solution of the corresponding subproblem is generated. Solution thus obtained is then transferred to the main program for updating the values of r_{ij}'s. Listing of the program is given in Appendix B.

5.3 COMPUTATIONAL PERFORMANCE

Performance of the algorithm is studied in terms of the number of subproblems solved. Information about CPU time is also collected but is not considered as an important index. Randomly generated problems up to 900 nodes have been solved in reasonable amount of CPU time (3 minutes or less).

Main aim of the computational experimentation is to study the number of iterations as a function of problem parameter

max R, and also its variation with n and density of the underlying graph.

Test problems are generated randomly by varying problem parameters n, de and maxR. For each fixed value of n, de and max R, three randomly generated LP problems are solved. Average number of iterations and CPU time (without problem generation time) are noted.

To study the effect of max R, on the number of iterations required, we define the parameter : IR = maxR/m, where m is the number of edges in the graph, i.e., $m = \frac{n(n-1)}{2} \times de$. For n = 25, 50, 100, 200, 300, random test problems having underlying graph of low density, such that the number of edges present are n, 2n, 3n, are solved. For n = 50, 100, 150, 200, problems are also solved where the graphs are of little higher densities, in such a way that the number of edges, present are $\approx$ 10n. Value of maxR for each problem (i.e. for a fixed value of n, m/n) is selected such that IR = .5, 1,2,3,4,5 and 6 i.e., maxR = .5m, m, 2m, 3m, 4m, 5m and 6m. Computational results are shown in Tables 5.1, 5.2, 5.3, 5.4 and 5.5. Table 5.1 shows the number of subproblems (iterations) solved for various values of IR. Here it is observed that the number of iterations becomes almost constant when IR is increased beyond 2, for a specified value of n and m.

Table 5.2 shows NR = (number of iterations/no of nodes) for varying values of n and max R.

Fig. 5.3(a), shows the variations of NR with IR. It can be observed that these ratios (NR)are very close to each other and are almost independent of the number of nodes. Let I(n) denote the maximum value of NR for n nodes. This can be taken as number of iterations/number of nodes when IR = 2, Variations of I(n) with n, for different values of m/n are plotted in fig. 5.3(b) It can be observed that for a fixed m/n, I(n) is constant or is independent of n. Further more the value of I(n) is increasing with the increase in m/n, but even for densed graphs it is approximately 2. Thus we can conclude that the number of iterations are of O(n). In fact, such an observation can be explained by the fact that the number of iterations are governed by the in-differenceness of r_{ij}'s and hence really are not dependent upon the magnitude of maxR, but on the distinct number of r_{ij}'s in the problem.

To show further the behaviour of I(n) for dense graphs, random problems are generated for n = 50, 100, 150 and 200 with $\frac{m}{n} \approx 10$.

Table 5.3 shows the variation of NR with max R. For these problems also above observations are confirmed.

Table 5.4 shows average computation time in CPU(seconds) for randomly generated problems of size n = 25 to 800, when m is taken to be 2n and max R = 2m. It is observed that even such large problems can be solved within a reasonable amount of time (less than 3 minutes).

5.4 CONCLUSION

With the above theoretical and computational experiences we are now able to conclude the following, regarding algorithm (3.4):

(a) For any LP, size of the first problem is very small and the number of nodes in successive iterations are nondecreasing. During the last few iterations subproblems on all the n-nodes may have to be solved. Therefore on an average for the sub-problems LP $[G_i; r_i c]$, we can take G_i to have n/2 nodes and $\frac{n(n-1)}{2} \times \frac{de}{2}$ edges.

(b) For different values of n and de, number of iterations generally varies between n and 2n for large values. Therefore the average number of iterations can be taken to be $\frac{3}{2}$ n.

Although the number of iterations i.e., the number of maximum-flow problems solved, are observed to be approximately $\frac{3n}{2}$, total number of generated subproblems, which depends upon r_{ij}'s, may be much larger than the number of iterations (i.e. the number of maximum flow problems). Since the actual computational effort needed for generating a subproblem and comparing it with the preceeding subproblem, is too small in comparision to the computational effort required to solve it, it will not be bad to state that the complexity of the algorithm is bounded polynomially by a function of the problem size (no of nodes).

Table No. 5.1

For number of iterations

n	m/n	Max R/m						
		.5	1	2	3	4	5	6
25	1	9	12	15	1	16	16	17
"	2	13	17	21	21	22	23	23
"	3	16	21	23	25	26	25	26
50	1	18	24	32	33	32	33	34
"	2	31	37	44	44	47	46	46
"	3	35	41	50	50	49	52	53
100	1	37	49	65	68	68	72	70
"	2	54	70	86	89	86	88	89
"	3	68	90	99	94	104	112	110
200	1	75	105	127	132	132	135	134
"	2	115	145	165	177	181	178	181
"	3	146	169	200	198	202	215	215
300	1	116	163	200	198	209	218	218
"	2	171	210	252	260	264	274	272
"	3	204	256	290	301	306	305	306

Table 5.2

[Number of iterations/n]

n	m/n	[Max R/m] .5	1	2	3	4	5	6
25	1	.36	.48	.66	.63	.64	.64	.68
"	2	.52	.65	.84	.86	.88	.92	.92
"	3	.63	.84	.93	.95	1.02	.99	.93
50	1	.36	.48	.64	.66	.66	.68	.68
"	2	.53	.77	.89	.88	.94	.92	.92
"	3	.69	.81	.99	.99	.96	1.02	1.02
100	1	.37	.49	.65	.68	.68	.72	.70
"	2	.54	.78	.86	.90	.96	.88	.90
"	3	.69	.90	.99	.93	1.05	1.02	.99
200	1	.38	.53	.63	.66	.66	.68	.67
"	2	.58	.72	.82	.88	.90	.88	.90
"	3	.72	.84	1.02	.99	1.02	1.00	1.03
300	1	.39	.54	.66	.66	.69	.73	.75
"	2	.56	.70	.89	.86	.88	.91	.90
"	3	.69	.84	.99	.99	1.02	1.02	1.02

Table 5.3

[Number of iterations/n]

n	m/n	[Max R/m]			
		.5	1	2	3
50	10	1.1	1.2	1.25	1.4
100	10	1.1	.9	1.29	1.3
150	10	1.0	1.2	1.24	1.25
200	10	.9	1.13	1.3	1.3

Table 5.4

For CPU time

n	Iterations	CPU time(seconds)
25	21	.287
50	44	.94
100	86	3.505
200	165	13.144
300	252	36.192
400	375	68.52
500	459	100.95
600	497	118.23
800	648	195.82

Table 5.5

[Number of iterations]

n	density	Max R/m .5	1	2	3	4
25	.50	23	25	29	30	30
"	.75	25	28	36	34	36
50	.50	55	60	65	67	66
"	.75	54	62	71	72	70
75	.50	87	107	110	110	109
"	.75	87	111	111	118	115

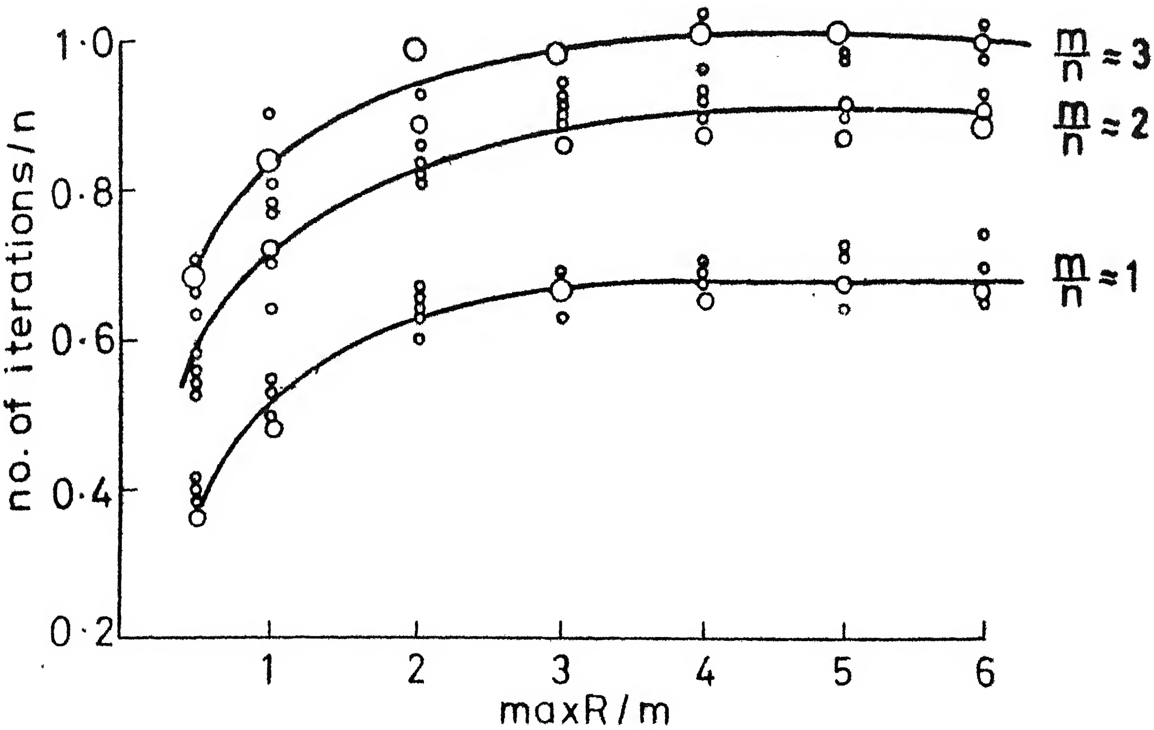


Fig. 5.3 (a)

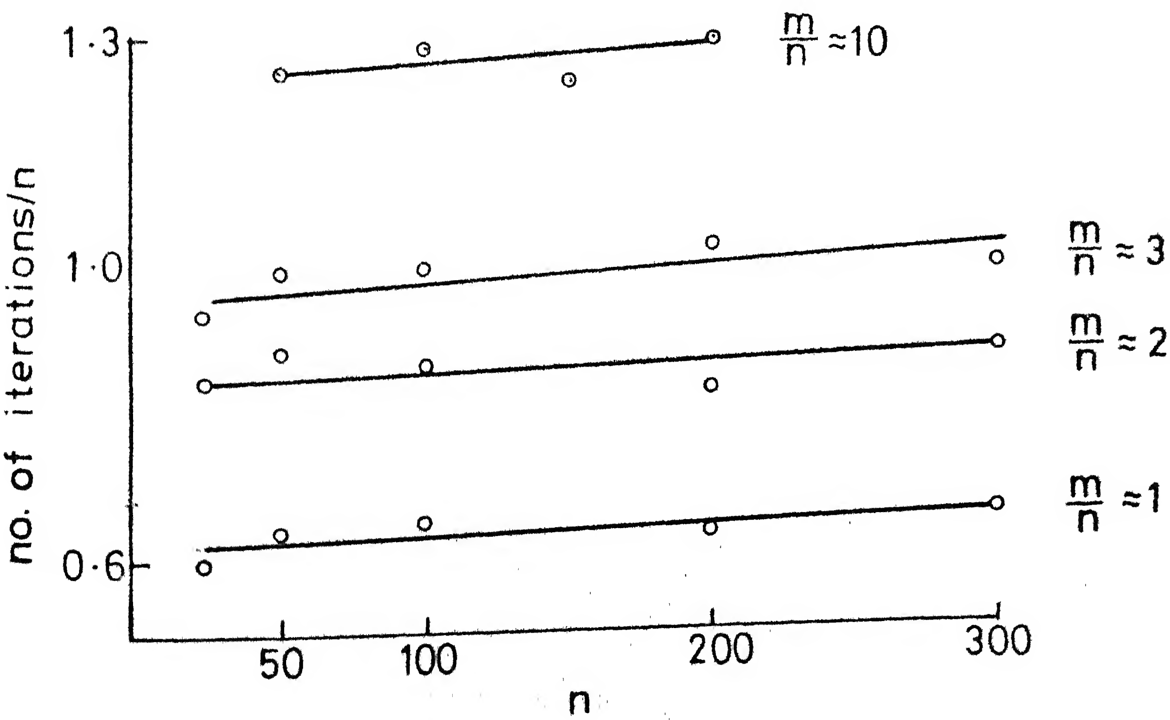


Fig. 5.3 (b)

CHAPTER VI

OPTIMAL COMMUNICATION SPANNING TREES

6.1 INTRODUCTION

Hu [27] has discussed the optimal communication spanning tree problem for a given set of requirements among n nodes. He has also suggested an algorithm [27] for constructing such an optimal communication spanning tree. Here we extend the above problem to the following three cases :

(i) Construction of an optimal communication spanning tree such that all the nodes of a given set S are outer nodes.

(ii) Construction of a new optimal communication spanning tree with the help of an existing one, when some of the requirements have been increased.

(iii) Construction of an optimal communication spanning tree such that some specified edges are present in the communication tree.

In the last section we suggest a minor modification to Hu's algorithm [27], for constructing a cut tree. This in some cases reduces, its total computational effort.

6.2 NOTATIONS AND DEFINITIONS

Definitions are collected from Hu's book [26].

G : Is a complete undirected graph on n nodes. These nodes may represent cities which need to communicate with each other. An edge joining nodes i and j will be denoted by an unordered pair (i,j).

r_{ij} : Is a nonnegative integer number assigned to every pair (i,j), such that $r_{ij} = r_{ji}$ where $i \neq j$; and $i = 1,2,\ldots,n$, $j = 1,2,\ldots,n$. It represents the requirement (e.g. number of approximate telephone lines needed) between nodes i and j.

d_{ij} : A nonnegative number assigned to every pair (i,j). This represents the distance between the stations represented by the nodes i and j in G. We will here assume that $d_{ij} = d_{ji} = 1$, $\forall\ (i,j) \in G$.

$G(r_{ij})$: Graph G with capacity r_{ij} on its arc (i,j); $\forall\ (i,j) \in G$.

Outer node of a tree : A node, whose degree is one, in the tree.

$C(T)$: Total cost of communication on the spanning tree T. There is a unique path from node i to node j in T. Length of this path is the sum of the distances between its adjacent nodes in the path. Cost of communication for a pair of nodes i and j is equal to r_{ij} multiplied by the distance from i to j in T. Summing up these communication costs for all the $\binom{n}{2}$ pairs, we get the total cost of communication on T.

$R(X,\bar{X})$: Is the capacity of a cut $(X,\bar{X})$, seperating two nodes i_1 and i_2. Here X is a subset of N (node set of G) containing i_1, and $\bar{X}$ is the complement of X with respect to N, and $R(X,\bar{X}) = \sum_{i \in X} \sum_{j \in \bar{X}} r_{ij}$

Since $r_{ij} = r_{ji}\ \forall\ (i,j) \in G$,

$R(X,\bar{X}) = R(\bar{X},X)$.

$\bar{f}_{pq}$: If an edge (i,j) having capacity r_{ij} is taken as the union of r_{ij} edges, each having capacity one in G then the maximum number of arcwise disjoint paths between the two nodes p and q in G will be denoted by $\bar{f}_{pq}$.

Minimum cut seperating p and q :

A cut $(X,\bar{X})$, such that $p \in X$ and $q \in \bar{X}$ is called a minimum cut seperating nodes p and q, if $R(X,\bar{X})$ is the least among the capacities of all the cuts seperating p and q.

Cut tree : A cut tree T of a graph $G(r_{ij})$ is a spanning tree having nonnegative integers v_{ij} corresponding to each of its edges (i,j), such that the followings are true :

(a) : If an edge (i,j) with the corresponding number v_{ij} is removed, nodes of $G(r_{ij})$ get partitioned into two sets X and $\bar{X}$, and

$$v_{ij} = R(X,\bar{X}).$$

Also this cut $(X,\bar{X})$ is a minimum cut seperating nodes i and j in $G(r_{ij})$.

(b) : Maximum flow $\bar{f}_{pq}$ from p to q in $G(r_{ij})$ is equal to

$$\min \{v_{pa}, \ldots, v_{ij}, \ldots, v_{tq}\}$$

where the edges of $\{(p,a),\ldots,(i,j),\ldots,(t,q)\}$ form the unique path from p to q in T.

$N^T(p)$: (Neighbourhood of p in the tree T).

$$N^T(p) = \{q \in N \mid \exists \text{ an edge } (p,q) \text{ in } T\}.$$

where N is node set of G.

v_{ij}^A : (i) For an edge (i,j) of A, where A is a spanning tree on n nodes, v_{ij} is the capacity on the arc (i,j), let (X_1,X_2) be a cut set seperating i and j in A, then

$$v_{ij}^A = R(X_1,X_2).$$

(ii) If $(i,j) \notin A$, v_{ij}^A is the maximum flow from i to j in the graph $A(v_{pq})$ where v_{pq} is taken as the capacity of the edge (p,q) in A, $\forall$ $(p,q) \in A$.

6.3 SOME PROPERTIES OF THE OPTIMAL COMMUNICATION SPANNING TREES

Hu [27] has shown that a cut tree of $G(r_{ij})$ is an optimal communication spanning tree, for the set of requirements $\{r_{ij}\}$. Here we will show that the converse of the above is also true i.e. any optimal communication spanning tree for the set of requirements $\{r_{ij}\}$ is also a cut tree for the graph $G(r_{ij})$, where G is a complete graph, and r_{ij}'s are taken as the capacities of its arcs (edges). In order to prove this, we first report the following observation :

Observation (6.3.1) : Capacities of the cuts given by the edges of an optimal communication spanning tree are the (n-1) maximum flows present in the graph $G(r_{ij})$.

Proof : Let A be a cut tree for $G(r_{ij})$ and let B be an optimal communication spanning tree. Let V_1 be an (n-1) dimensional vector whose components are the capacities of the cuts given by the edges of A.

$$V_1 = \begin{bmatrix} v^A_{12} \\ v^A_{ij} \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

where v^A_{ij} is the capacity of the cut given by an edge (i,j) of A in $G(r_{ij})$. We will now use the following result given by Adolphson and Hu [1] :

"Given two spanning trees of a network G we shall refer to one as the 'red' spanning tree and the other as the 'blue' spanning tree. For any blue edge (p,q) there is a red edge (p_1,q_1) such that (p_1,q_1) is one of the red edges which form the unique path from p to q in the red spanning tree.

Furthermore, a one to one mapping f_1 among the blue edges and red edges can be established which satisfies the above condition".

Using the above mapping 'f_1' from the set of edges of A to the set of edges of B, we can construct an $(n-1)$ dimensional vector V_2 where a component of V_2 is the capacity of the cut given by the edge $f_1(i,j)$ of B. We claim that the vectors V_1 and V_2 are the same. We prove it as follows : Since A and B both are optimal communication spanning trees, total costs of communication on A and B are equal. Hu [27] has shown that the total cost of communication on a spanning tree is equal to the sum of the cut-capacities given by its $(n-1)$ edges. Therefore

$$eV_1 = eV_2$$

where e is the (n-1)-dimensional summation vector. Suppose, a component of V_2 is greater than the corresponding component of V_1, then there must exist a component of V_2 which is strictly smaller than its corresponding component in V_1. Let this component of V_1 be $v^A_{r_1 r_2}$ i.e.

$$v^B_{f(r_1,r_2)} < v^A_{r_1 r_2} \qquad 6.3(a)$$

where $f(r_1,r_2) = p'q'$ such that $(p',q') \equiv f_1(r_1,r_2)$. Edge $f_1(r_1,r_2)$ is in the path from r_1 to r_2 in B, therefore it gives a cut set $(X,\bar{X})$ seperating nodes r_1 and r_2 in $G(r_{ij})$ and capacity of this cut is $v^B_{f(r_1,r_2)}$. Since A is a cut tree for $G(r_{ij})$, therefore $v^A_{r_1 r_2}$ is the capacity of the minimum cut seperating nodes r_1 and r_2 in $G(r_{ij})$. Hence the inequality 6.3(a) gives a contradiction. Thus V_1 and V_2 are not different.

Since A is a cut tree, components of V_1 are all the (n-1) maximum flows present in the graph $G(r_{ij})$. Hence the edges of any optimal communication spanning tree give all the (n-1) maximum flows for $G(r_{ij})$.

Theorem (6.3.1) : Any optimal communication spanning tree B for the set of requirements $\{r_{ij}\}$, is a cut tree for the associated graph $G(r_{ij})$.

Proof : By the observation (6.3.1) edges of B correspond to the (n-1) minimum cuts seperating all the $\binom{n}{2}$ pair of nodes of $G(r_{ij})$. Hence an edge of B represents a minimum cut seperating a certain pair of nodes of $G(r_{ij})$.

Now by the theorem (9.2) [26] proved by Hu, this spanning tree B will have all the properties of a cut tree.

Hence B is a cut tree for $G(r_{ij})$.

□

6.4 OPTIMAL COMMUNICATION SPANNING TREE HAVING ALL THE NODES OF A SPECIFIED SET, AS ITS OUTER NODES

We now present an algorithm for constructing a spanning tree such that it contains all the nodes of a specified subset S of N, as its outer nodes, and the total cost of communication on it, is also minimum among such spanning trees.

Let $S = \{ s_1, s_2, \ldots, s_k, s_{k+1}, \ldots, s_b \}$ be this specified set.

For a spanning tree B(p) of $G(r_{ij})$, where p is a integer number.

$$S_o(p) = \{ s_q \mid s_q \in S \text{ and it is an outer node of } B(p) \}.$$

For a node s_{p+1} of $S - S_o(p)$, define

$$NB^{B(p)}(s_{p+1}) = \{ q \in G \mid (s_{p+1}, q) \in B(p) \text{ and } q \notin S_o(p) \}.$$

Let $B(0) = A$, where A is a cut tree for $G(r_{ij})$,

and let, $S_o(0) = \{ s_{k+1}, \ldots, s_b \}$. where $|S| = b$

Starting with an optimal communication spanning tree B(0) for $\{r_{ij}\}$ we will construct a sequence of spanning trees B(1),...,B(k) such that

$$S_o(p) \subset S_o(p+1),$$

$$|S_o(p+1)| = |S_o(p)| + 1$$

and $\quad |S_o(k)| = b$ and $S_o(k) = S$.

For every p, $p = 1,\ldots,k$, spanning tree $B(p)$ will be obtained from the spanning tree $B(p-1)$ by the following algorithm :

Algorithm (6.4) :

Step 0 : Let $s_p \in S$ and it is not an outer node of $B(p-1)$.

Find $$v^{B(p-1)}_{s_p q_p} = \max_{j \in NB^{B(p-1)}(s_p)} \left\{ v^{B(p-1)}_{s_p j} \right\} \qquad 6.4(a)$$

Step 1 : Let $j \in N^{B(p-1)}(s_p)$. Add an edge (j,q_p) and delete the edge (s_p,j) from $B(p-1)$.

After each repetition of step one, we get a spanning tree and after $|N^{B(p-1)}(s_p)|$ repetitions, a spanning tree, in which the node s_p is also an outer node, is obtained. We will denote this spanning tree by $B(p)$.

By definition $NB^{B(p-1)}(s_p) \cap S_o(p-1) = \phi$ and $q_p \in NB^{B(p-1)}(s_p)$, it is easy to see that,

$$S_o(p-1) \subset S_o(p)$$

and $$|S_o(p)| = |S_o(p-1)| + 1.$$

For example let $B(p-1)$ be,

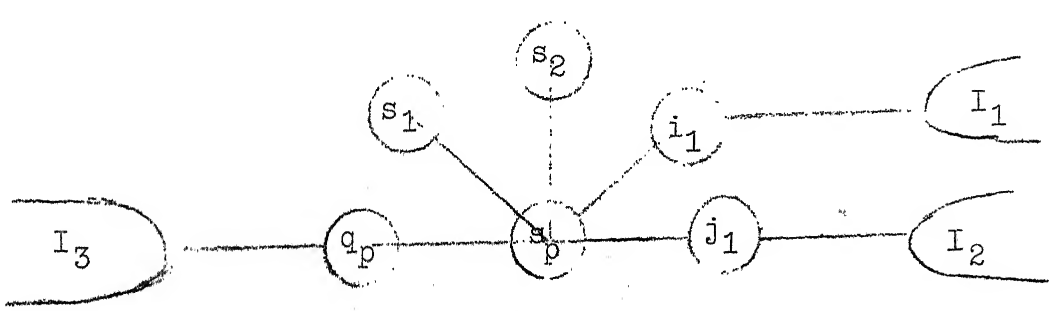


Figure 6.4(1)

where I_1, I_2 and I_3 denote the parts of $B(p-1)$.

Let

$$S_o(p-1) = \{s_1, s_2, \ldots, s_{p-1}, s_{k+1}, \ldots, s_b\}$$

$$NB^{B(p-1)}(s_p) = \{q_p,\ i_1,\ j_1\}$$

and $$v_{s_p q_p}^{B(p-1)} = \max\left\{v_{s_p q_p}^{B(p-1)},\ v_{s_p i_1}^{B(p-1)},\ v_{s_p j_1}^{B(p-1)}\right\}.$$

Then by the above algorithm (6.4), B(p) will be

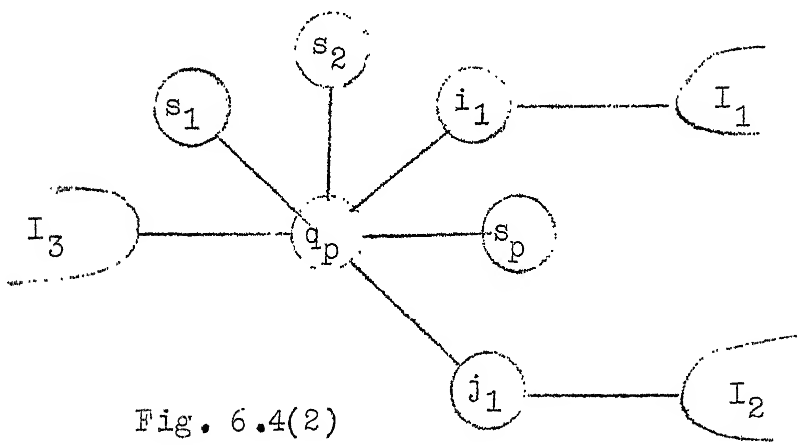


Fig. 6.4(2)

Now

$$S_o(p) = \{s_1, s_2, \ldots, s_{p-1}, s_p, s_{k+1}, \ldots, s_b\}.$$

Property (6.4) : From the construction of B(p), it is easy to see that for every edge (i, j) of B(p) which was also in B(p-1) but not incident upon s_p there, following is true :

$$v_{ij}^{B(p)} = v_{ij}^{B(p-1)}. \qquad 6.4(b)$$

eorem (6.4.1) : For any two nodes p and q such that neither or q belongs to $S_o(k)$, value of the minimum cut given by the

tree $B(k)$ is also a minimum cut seperating nodes p and q in $G(r_{ij})$.

Proof : We prove it by induction. Let $B(0) = A$ where A is an optimal communication spanning tree, for $G(r_{ij})$. By theorem (6.3.1) A will be a cut tree for $\{r_{ij}\}$. Therefore above theorem is true for $B(i)$, when $i = 0$. Let the theorem be true for $B(i-1)$, $i \geq 1$ we will now show that it will be true for $B(i)$ also.

Select arbitrary two nodes of G, say p and q, such that neither p nor q is in $S_0(i)$.

Let $(p, i_1, i_2, \ldots, i_r, q)$ be the unique path from p to q in $B(i-1)$ and let s_i be the node which is made an outer node in $B(i)$. Consider the following cases :

Case 1 : s_i does not lie on this path.

Case 2 : s_i does lie on this path.

Case 1 : By construction of $B(i)$ from $B(i-1)$, it is easy to see that the unique path from p to q in $B(i)$ is the same, as it was in $B(i-1)$.

$$\therefore \quad v_{pq}^{B(i)} = \min \left\{ v_{pi_1}^{B(i)},\ v_{i_1 i_2}^{B(i)}, \ldots,\ v_{i_r q}^{B(i)} \right\}.$$

By 6.4(b) it can also be written as

$$= \min \left\{ v_{pi_1}^{B(i-1)}, \ldots, v_{i_r q}^{B(i-1)} \right\}$$

$$= v_{pq}^{B(i-1)}.$$

Since neither p nor q is in $S_o(i-1)$ and the theorem is assumed to be proved for $k \geq i-1$, $v_{pq}^{B(i-1)}$ is the capacity of the minimum cut seperating p and q in $G(r_{ij})$. Hence $v_{pq}^{B(i)}$ is also the capacity of the minimum cut seperating nodes p and q in $G(r_{ij})$.

Case 2 : Let q_i be the node adjacent to s_i in B(i). Here again two more cases are possible :

(i) q_i does not lie on this path.

(ii) q_i does lie on this path.

Case 2(i) : In B(i-1), let the path from p to q be

$$(p, i_1, \ldots, i_o, s_i, j_o, \ldots, j_r, q) .$$

For example let

B(i-1) :

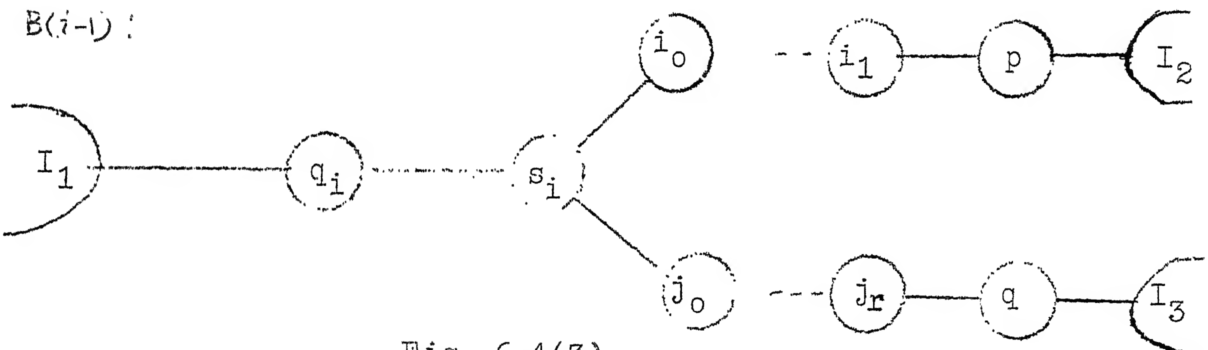


Fig. 6.4(3)

Since

$$v_{pq}^{B(i-1)} = \min \left\{ v_{pi_1}^{B(i-1)}, \ldots, v_{i_o s_i}^{B(i-1)}, v_{s_i j_o}^{B(i-1)}, \ldots, v_{j_r q}^{B(i-1)} \right\}$$

In B(i) path from p to q will be $(p, i_1, \ldots, i_o, q_i, j_o, \ldots, j_r, q)$.

Therefore

$$v_{pq}^{B(i)} = \min \left\{ v_{pi_1}^{B(i)}, \ldots, v_{i_o q_i}^{B(i)}, v_{q_i j_o}^{B(i)}, \ldots, v_{j_r q}^{B(i)} \right\} .$$

B(i) :

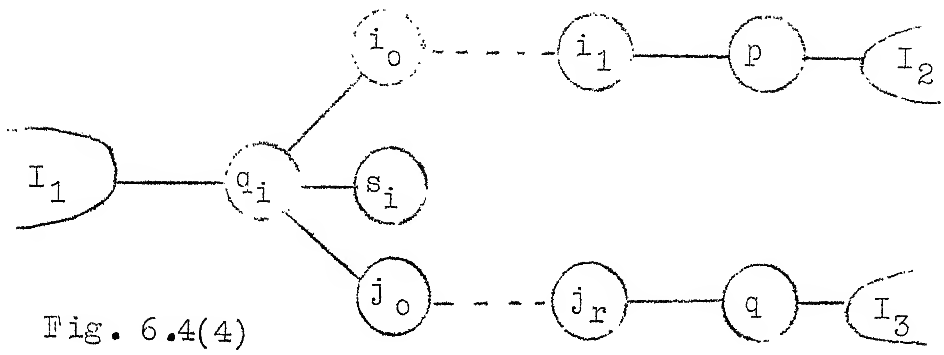


Fig. 6.4(4)

By construction of B(i) from B(i-1), it is also easy to see that,

$$v^{B(i)}_{i_o q_i} = v^{B(i-1)}_{i_o s_i}$$

and

$$v^{B(i)}_{j_o q_i} = v^{B(i-1)}_{j_o s_i}.$$

Hence by 6.4(b) we can say,

$$v^{B(i)}_{pq} = v^{B(i-1)}_{pq}.$$

Since the theorem is assumed to be true for B(i-1) and neither p nor q is in $S_o(i-1)$, $v^{B(i-1)}_{pq}$ and therefore $v^{B(i)}_{pq}$ is the capacity of the minimum cut seperating nodes p and q in $G(r_{ij})$.

Case (ii) : In B(i-1), let the path from p to q be $(p, i_1, \ldots, i_o, s_i, q_i, j_o, \ldots, j_r, q)$. Path from p to q in B(i) will therefore be $(p, i_1, \ldots, i_o, q_i, j_o, \ldots, j_r, q)$.

For example if B(i-1) :

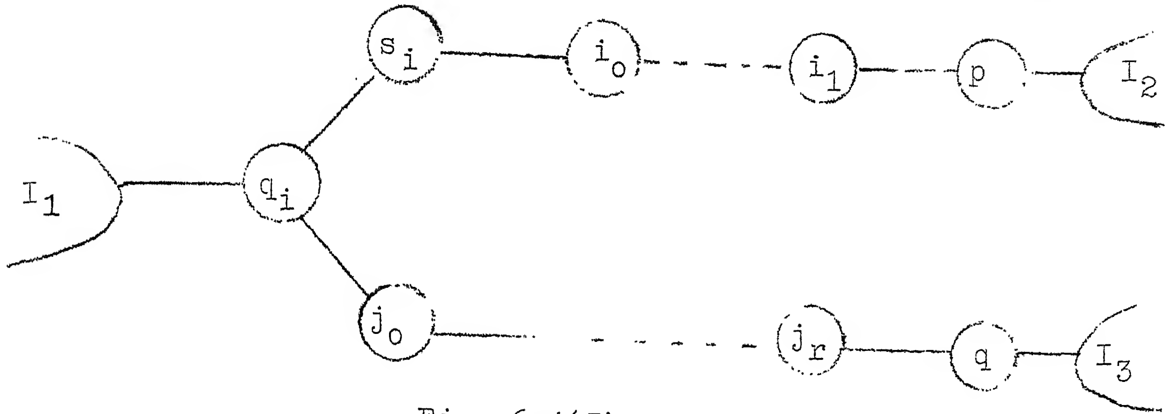


Fig. 6.4(5)

Then B(i) will be

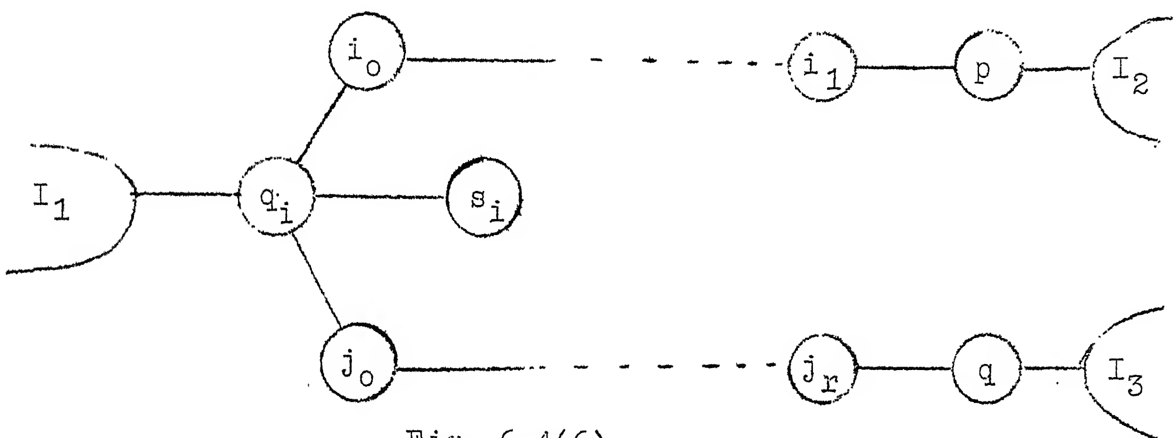


Fig. 6.4(6)

By the choice of q_i (from equation 6.4(a)) we know that

$$v_{s_i q_i}^{B(i-1)} \geq v_{i_o s_i}^{B(i-1)} .$$

Hence $v_{pq}^{B(i-1)}$ can also be written as,

$$v_{pq}^{B(i-1)} = \min\left\{ v_{p i_1}^{B(i-1)}, \ldots, v_{i_o s_i}^{B(i-1)}, v_{q_i j_o}^{B(i-1)}, \ldots, v_{j_r q}^{B(i-1)} \right\} .$$

Also $v_{pq}^{B(i)} = \min\left\{ v_{p i_1}^{B(i)}, \ldots, v_{i_o q_i}^{B(i)}, v_{q_i j_o}^{B(i)}, \ldots, v_{j_r q}^{B(i)} \right\}$.

By construction of B(i) from B(i-1) it is easy to see that

$$v_{i_o q_i}^{B(i)} = v_{i_o s_i}^{B(i-1)}.$$

By 6.4(b), cut capacities for the other common edges of the two paths from p to q in B(i-1) and B(i) respectively, are the same. Hence

$$v_{pq}^{B(i-1)} = v_{pq}^{B(i)}.$$

Hence $v_{pq}^{B(i)}$ is the capacity of the minimum cut seperating nodes p and q in $G(r_{ij})$.

Since the above theorem has been proved for an arbitrary i, Therefore it will be true for i = k also.

□

We will now show that B(k) is the required optimal communication spanning tree.

Theorem (6.4.2) : Total cost of communication on the spanning tree B(k) is less than or equal to the total cost of communication over any other spanning tree of $G(r_{ij})$ which has all the nodes of S as its outer nodes.

Proof : Proof is by contradiction. Let C_o be a spanning tree having all the nodes of S as outer nodes and let the total cost of communication on C_o be strictly less than the total cost of communication on B(k). For every $s_j \in S$, let t_j and v_j be its neighbours in C_o and B(k) respectively.

Now consider the trees B' and C_o', where

$$B' = B(k) - \bigcup_{s_i \in S} \{(s_i, v_i)\}$$

and $$C_o' = C_o - \bigcup_{s_i \in S} \{(s_i, t_i)\}$$

i.e. B' is obtained from $B(k)$ by removing all the edges (s_i, v_i) where $s_i \in S$. Similarly C_o' is obtained from C_o by removing from it all the edges (s_i, t_i) where $s_i \in S$.

We now use Adolphson's [1] mapping 'f_1' from the set of edges of B' to set of edges of C_o'. Node sets of B and C_o' are the same. For every edge (p_i, p_j) of B', there is an edge $f_1(p_i, p_j) = (q_i, q_j)$ of C_o', which is in the unique path from p_i to p_j in C_o'. $f_1(p_i, p_j)$ also lies in the unique path from p_i to p_j in C_o. Therefore the edge $f_1(p_i, p_j) = (q_i, q_j)$ in C_o' gives a cut seperating p_i and p_j in $G(r_{ij})$. By theorem (6.4.1) $v^{B(k)}_{p_i p_j}$ is the capacity of the minimum cut seperating p_i and p_j in $G(r_{ij})$.

Therefore $$v^{B(k)}_{p_i p_j} \leq v^{C_o}_{q_i q_j} \quad \forall (p_i, p_j) \in B'$$

Hence $$\sum_{(p_i, p_j) \in B'} v^{B(k)}_{p_i p_j} \leq \sum_{(q_i, q_j) \in C_o'} v^{C_o}_{q_i q_j}$$

Now for any node s_i of S,

$$v^{B(k)}_{s_i v_i} = \sum_{p \in G} r_{s_i p}$$

and $$v^{C_o}_{s_i t_i} = \sum_{p \in G} r_{s_i p}.$$

Therefore $\sum_{s_i \in S} v^{B(k)}_{s_i v_i} = \sum_{\substack{s_i \in S \\ (s_i, t_i) \in C_o}} v^{C_o}_{s_i t_i}$

Thus

$$\sum_{(p_i,p_j) \in B'} v^{B(k)}_{p_i p_j} + \sum_{s_i \in S} v^{B(k)}_{s_i v_i} \leq \sum_{(s_i,s_j) \in C'_o} v^{C_o}_{s_i s_j} + \sum_{s_i \in S} v^{C_o}_{s_i t_i}$$

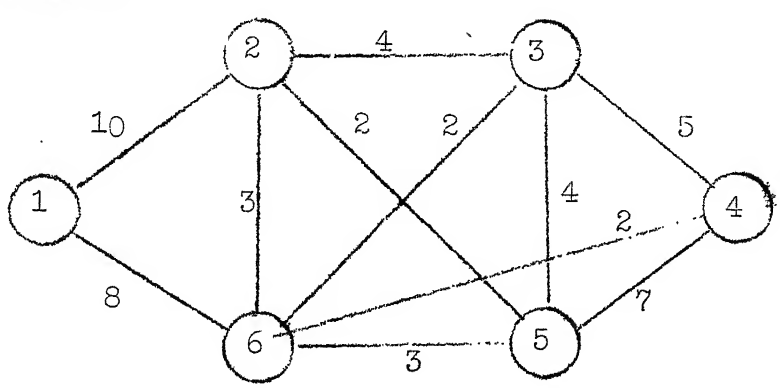
and hence

$$\sum_{(p_i,p_j) \in B(k)} v^{B(k)}_{p_i p_j} \leq \sum_{(s_i,s_j) \in C_o} v^{C_o}_{s_i s_j} \qquad 6.4(c)$$

Hu [27] has shown that the total cost of communication over a spanning tree is sum of the capacities of cuts given by its edges. Thus the inequality 6.4(c) gives a contradiction. Hence B(k) is the required optimal communication spanning tree.

□

Example (6.4) : This example is borrowed from Hu [27]



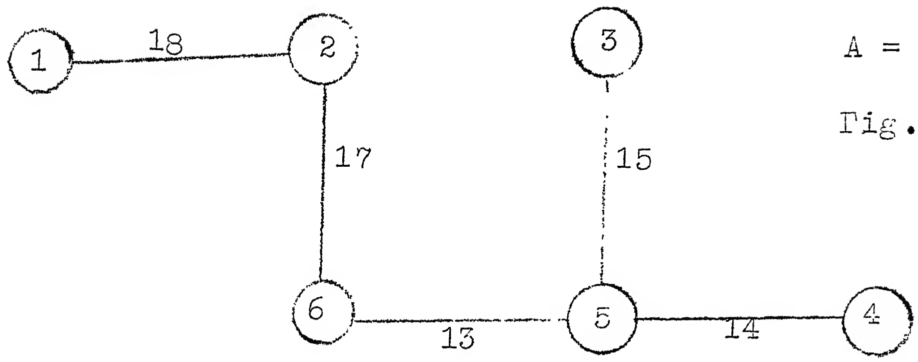
G(r_{ij})

Fig. 6.4(7)

(Edges having zero capacities are not shown).

Optimal communication spanning tree A obtained by Hu for the set

of requirements $\{r_{ij}\}$ is



A = B(0)

Fig. 6.4(8)

Let the specified set S whose nodes are required to be outer nodes in the communication spanning tree, be = {2,5,6}.

$$S_0(0) = \phi \ .$$

Let us first select node 2 to make it an outer node i.e.

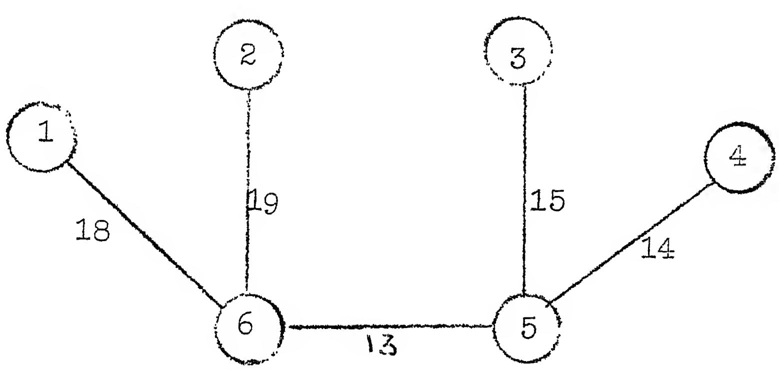
$$s_1 = 2$$

$$v_{21}^{B(0)} = \max_{i \in NB^{B(0)}(2)} \left\{ v_{2i}^{B(0)} \right\}$$

$$= 18$$

Hence $q_1 = 1$.

Therefore optimal communication spanning tree having node 2 as an outer node, is



B(1)

Fig. 6.4(9)

$$S_o(1) = \{2\}$$

node 5 of S is not in $S_o(1)$.

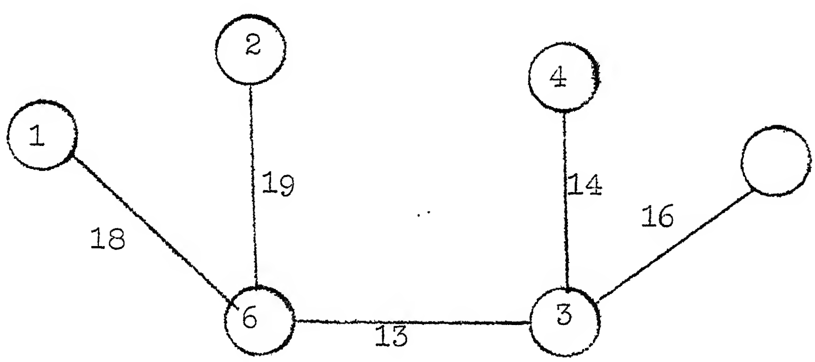
Now we select the node 5 to make it an outer node i.e. $s_2 = 5$.

$$v_{53}^{B(1)} = \max_{i \in NB^{B(i)}(5)} \left\{ v_{5i}^{B(1)} \right\}$$

$$= 15.$$

Therefore $\quad q_2 = 3.$

Hence following is the optimal communication spanning tree having node 2 and node 3 as its outer nodes :



B(2)

Fig. 6.4(10)

$$S_o(2) = \{2,5\}.$$

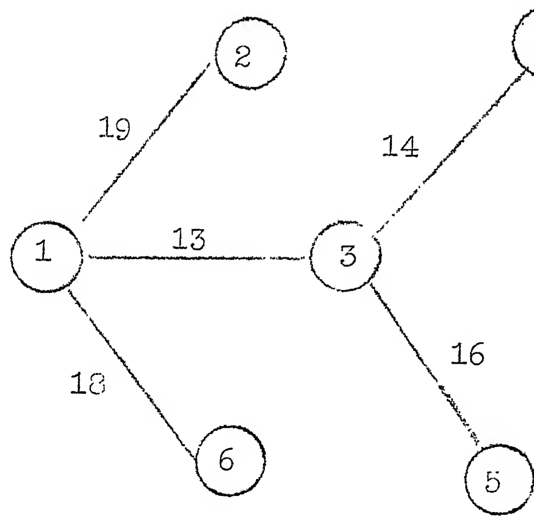
Now we will make node 6, an outer node, i.e. $s_3 = 6$

$$v_{61}^{B(2)} = \max_{i \in NB^{B(2)}(6)} \left\{ v_{6i}^{B(2)} \right\}$$

$$= 18.$$

Therefore $\quad q_3 = 1.$

Hence B(3) is,



$S_o(3) = \{2,5,6\} = S.$

Thus B(3) is the required optimal communication spanning tree.

6.5 OPTIMAL COMMUNICATION SPANNING TREE FOR THE UPDATED SET OF REQUIREMENTS

In this section an algorithm for constructing an optimal communication spanning tree for the updated set of requirements $\{r'_{ij}\}$ with the help of an optimal communication spanning tree A for the old set of requirements $\{r_{ij}\}$, is presented.

Suppose the requirements $\{r_{ij}\}$ have been updated to $\{r'_{ij}\}$ as follows :

(1) Requirement between nodes p_1 and p_2 is increased by δ_1 (δ_1 is a positive integer).

(2) Requirement between nodes q_1 and q_2 is increased by δ_2 (δ_2 is a positive integer).

.
.
.

(r) Requirement between nodes z_1 and z_2 is increased by δ_r (δ_r is a positive integer).

Let $E_o = \{(p_1,p_2), (q_1,q_2),\ldots,(z_1,z_2)\}$ be the set of all the pair of nodes for which requirements have been increased.

We will first find out those parts of the tree A, which can be made to be present in an optimal communication spanning tree for the updated set of requirements $\{r'_{ij}\}$.

Let $N(E_o) = \{i \mid (i,j) \in E_o$ for some $j\}$

Thus $N(E_o) = \{p_1, p_2, q_1, q_2, \ldots, z_1, z_2\}$.

Let T_{p+1} be the smallest connected subgraph of the given optimal communication spanning tree A which contains all the nodes of $N(E_o)$. Let E_1 be the set of all the edges of A which are incident upon a node of T_{p+1} but are not present in T_{p+1}. Deletion of all the edges of E_1 partitions the spanning tree A, into several components say $T_1, T_2, \ldots, T_{p+1}$. Let (t_i, p_i) be an edge of A which joins T_i and T_{p+1} where $t_i \in T_i$ and $p_i \in T_{p+1}$.

Set E_1 can also be written as :

$E_1 = \{(t_i, p_i) \mid (t_i\ p_i) \in A$ where $t_i \in T_i,\ p_i \in T_{p+1}\}$.

It is easy to see that if a minimum cut (X_1, X_2) seperating nodes i_1 and j_1 in $G(r_{ij})$ does not contain any edge of E_o, then the cut (X_1, X_2) will be a minimum cut seperating nodes i_1 and j_1 in $G(r'_{ij})$ also. Using this we will now show that an optimal communication spanning tree for $\{r'_{ij}\}$ can be constructed, which contains all the edges of $T_1 \cup T_2 \cup \ldots T_p$. For constructing a cut tree for $G(r'_{ij})$ as suggested by Hu [26], we can start with an arbitrary pair of nodes. Let us first take the pair t_i, p_i. Edge (t_i, p_i) of A gives a minimum cut $(\{T_i\}, \{\bar{T}_i\})$ seperating nodes t_i and p_i in $G(r_{ij})$, where $\{T_i\}$ denotes the set

of all the nodes contained in T_i and $\{\bar{T}_i\}$ is the set of all the nodes of G which are not in $\{T_i\}$. This cut does not contain any edge of E_o, therefore it will be a minimum cut seperating t_i and p_i in $G(r'_{ij})$ also. After repeating it for every pair of nodes t_i and p_i of E_1, structure of the cut tree constructed so far, for $G(r'_{ij})$, will look like :

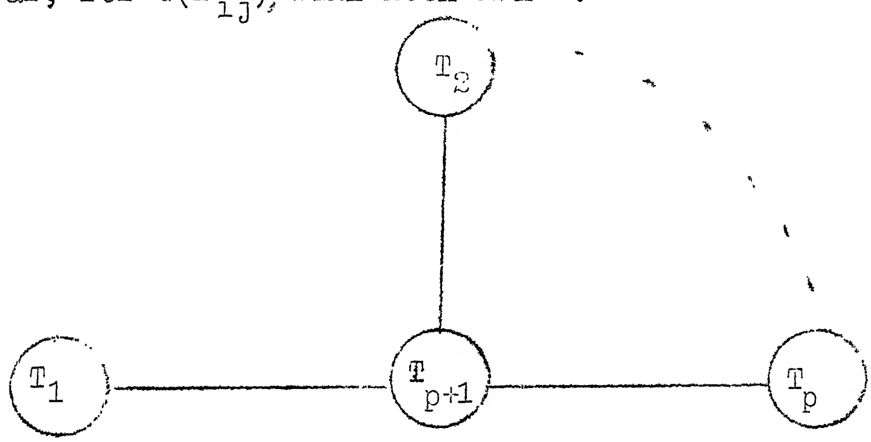


Fig. 6.5(1)

Cut given by any edge (i_1, i_2) of $\bigcup_{i=1}^{p} T_i$ does not contain any edge of E_o, therefore the same cut will be a minimum cut seperating nodes i_1 and i_2 in $G(r'_{ij})$ also. Hence all the edges of $\bigcup_{i=1}^{p} T_i$ can be put in a cut tree for $G(r'_{ij})$.

Now to obtain the complete cut tree for $G(r'_{ij})$, we have to add edges in $\{T_{p+1}\}$ only. This will be done by finding minimum cuts seperating different pair of nodes contained in $\{T_{p+1}\}$. These cuts will be obtained in a simpler graph G_o of G, which is obtained by condensing all the nodes of a T_i $(i \neq p+1)$ to a big node. Using the above results, a stepwise algorithm for constructing the required optimal communication spanning tree is given below :

Algorithm (6.5):

Step 0 : Find T_{p+1} in the given optimal communication spanning tree A. Find the minimum cut (X_1, X_2) seperating any two nodes contained in T_{p+1}, in the graph G_o.

Step 1 : Select a component, say X_1 containing more than one node of T_{p+1}. Find a minimum cut (Y_1, Y_2) seperating two nodes of T_{p+1}, which are contained in X_1. Now do branching from X_1 or X_2 according as $X_2 \subset Y_1$ or $X_2 \subset Y_2$ [26]. For example : if $X_2 \subset Y_1$, structure of the cut tree will look like

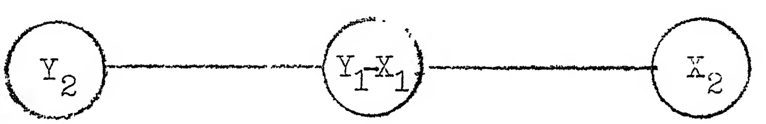


Fig. 6.5(2)

Repeat step 1, till each big node of the present cut tree contains not more than one node of T_{p+1}. Since all the cuts are determined in a simpler graph G_o, all the nodes of a T_i will be present in one big node of the present cut tree.

Step 2 : Let a big node X_o of the present cut tree, contain a node p_r of T_{p+1} and all the nodes of $\{T_{j_1}\} \cup \{T_{j_2}\} \cup \ldots \ldots \cup \{T_{j_t}\}$. Since we have already shown that $(\{T_j\}, \{\bar{T}_j\})$ is a minimum cut seperating t_j and p_j in $G(r'_{ij})$ also, therefore t_{j_i} of T_{j_i} will be connected to this node p_r of T_{p+1}, where t_{j_i} is

a node of T_{j_i} such that the edge (t_{j_i}, p_{j_i}) is in A for some p_{j_i} of T_{p+1}.

Thus an optimal communication spanning tree can be obtained by finding minimum cuts for (r-1) pair of nodes only, where r is the cardinality of the node set of T_{p+1}.

Corollary (6.5) : If all the edges of E_0 are the edges of A, then A itself will be an optimal communication spanning tree for the updated set of requirements $\{r'_{ij}\}$.

6.6 OPTIMAL COMMUNICATION SPANNING TREE HAVING SOME PRESPECIFIED EDGES

In the present section, we deal with the problem of constructing an optimal communication spanning tree, when it is required that, some specified pair of nodes should be adjacent. We will further assume that these specified pair of nodes do not form any cycle. Construction of such a spanning tree may be needed due to one of the following reasons :

(1) Requirement between such specified pair of nodes are of higher priority.

(2) Informations to be communicated between such pair of nodes are classified and therefore they should not pass through any other node.

Let $E_1 = \{(i,j) \mid i \text{ and } j \text{ are required to be adjacent}\}$.

We will show that a cut tree 'T' obtained for the graph $G(r'_{ij})$ is the required optimal communication spanning tree, where

$$r'_{ij} = r_{ij} + \left(\sum_{(i,j)\in G} r_{ij}\right) \times (n-1)\,, \text{ if } (i,j) \in E_1$$
$$= r_{ij} \qquad\qquad \text{otherwise.}$$

First we will show that this cut tree contains all the edges of E_1.

Theorem (6.6.1) : Every edge (i,j) of E_1 is present in T.

Proof : Let A be an arbitrary spanning tree on n nodes and having all the edges of E_1. Let B be a spanning tree on n nodes, not having atleast one edge of E_1. Let such an edge be (i_o, j_o). Length of the path from i_o to j_o in B is atleast two. Therefore the cost of communication from i_o to j_o on B is atleast

$$= 2 \times \left(\sum_{i,j} r_{ij}\right) \times (n-1)) + 2r_{i_o j_o}.$$

Total cost of communication over B is greater than

$$(|E_1| + 1) \times \left(\sum_{i,j} r_{ij} \times (n-1)\right) + R_1$$

where R_1 is the sum of the costs of communication on B with respect to the requirements $\{r_{ij}\}$.

Total cost of communication on A with respect to the set of requirements $\{r'_{ij}\}$ is

$$|E_1| \left(\sum_{i,j} r_{ij} \times (n-1)\right) + R_2$$

where R_2 is the sum of the costs of communication over A with respect to the set of requirements $\{r_{ij}\}$. It is easy to see that

$$R_2 < (\sum_{i,j} r_{ij}) \times (n-1)$$

Since $R_1 > 0$, total cost of communication over B is greater than the total cost of communication over A, with respect to the set of requirements $\{r'_{ij}\}$. Since T is a spanning tree having minimum cost of communication for the set of requirements $\{r'_{ij}\}$, it must have all the edges of E.

We will now show that this cut tree T for $G(r'_{ij})$ is the required optimal communication spanning tree. $\square$

<u>Theorem 6.6.2</u>. A cut tree T for the graph $G(r'_{ij})$ is of least total cost of communication, with respect to the set of requirements $\{r_{ij}\}$, among the spanning trees of G, and containing all the edges of E_1.

<u>Proof</u>. Take an optimal communication spanning tree A for $\{r_{ij}\}$, which contains all the edges of E_1. Now again we use Adolphson's mapping 'f_1' [1] from the set of edges of T to the set of edges of A.

Obviously $f_1(i,j) = (i,j) \quad \forall (i,j) \in E_1$.

Every edge (i_1,i_2) of T, such that $(i_1,i_2) \notin E_1$ gives a cut seperating nodes i_1 and i_2 in $G(r_{ij})$ also. This cut may or may not be a minimum cut seperating i_1 and i_2 in $G(r_{ij})$. From the definition of $\{r'_{ij}\}$, it is easy to see that the capacity of this cut in $G(r_{ij})$ is less than or equal to the capacity of any other cut in $G(r_{ij})$, seperating i_1 and i_2 and not containing any edge of E_1. Cut given by

the edge $f_1(i,j)$ of A, for every edge (i,j) of T, such that $(i,j) \notin E_1$ does not contain any edge of E_1. Let us denote the capacity of a cut given by an edge (p,q) of T in $G(r_{ij})$ by $\bar{v}_{pq}^{T}$. Hence

$$\bar{v}_{pq}^{T} = v_{pq}^{T} \quad \forall \ (p,q) \in T \ \text{ such that } \ (p,q) \notin E_1$$

And hence

$$\sum_{(p,q) \in T} \bar{v}_{pq}^{T} \leq \sum_{f_1(p,q)} v_{f(p',q')}^{A}$$

such that $(p,q) \notin E_1$

where; $f(p',q') = p'q'$; $(p',q') = f_1(p,q)$.

It is easy to see that in $G(r'_{ij})$ cut given by any edge (i_1,j_1) of T where $(i_1,j_1) \in E_1$ will be a minimum cut seperating i_1 and j_1 in $G(r_{ij})$ also, and hence

$\bar{v}_{i_1j_1}^{T} = \left[v_{i_1j_1}^{T} - \sum_{(i,j)\in G} r_{ij} \times (n-1) \right]$ will be the value of the minimum cut seperating i_1 and j_1 in $G(r_{ij})$.

Therefore $\sum_{(p,q)\in T} \bar{v}_{pq}^{T} < \sum_{(p,q)\in A} v_{f(p',q')}^{A}$ where $(p',q') = f_1(p,q)$, which gives a contradiction to the optimality of A.

Hence T is the required optimal communication spanning tree.

□

6.7 Modification to the Algorithm [26] for constructing a cut tree

Hu [26] has given an algorithm for constructing a cut tree of a graph $G(r_{ij})$. In this algorithm (n-1) maximum flow

problems are needed to be solved, where n is the number of nodes in the graph G. In this section we present some results which in some cases can be used for reducing the number of maximum flow problems required to be solved for constructing a cut tree.

We also specify an order in which nodes are to be selected for solving the corresponding maximum flow problems. This way of selection in some cases reduces the number of maximum flow problems to be solved. Since the selection of nodes in this specific order does not require any extra computational effort, Hu's [26] algorithm can be modified by this proposed order.

<u>Theorem (6.7.1)</u> Any node i_1 of N, such that,

$$\sum_{j\in G} r_{i_1 j} = \max_{i\in G} \left\{ \sum_{j\in G} r_{ij} \right\}$$

will be an inner node (i.e. its degree is greater than one) in at least one optimal communication spanning tree for $\{r_{ij}\}$.

<u>Proof</u>. Let T_1 be an optimal communication spanning tree for the set of requirements $\{r_{ij}\}$ such that i_1 is its outer node. Let $(i_1, i_2) \in T_1$

T_1:

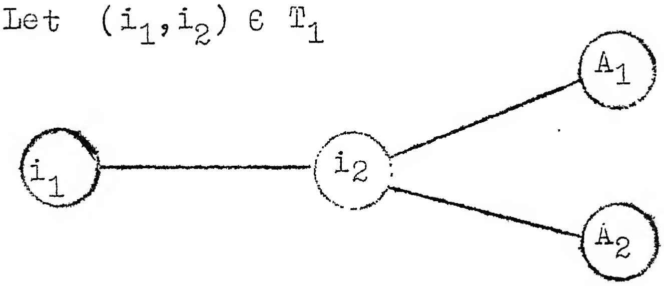


Fig. 6.7(1)

From T_1, let us construct a spanning tree T_2, just by interchanging the positions of i_1 and i_2.

T_2 :

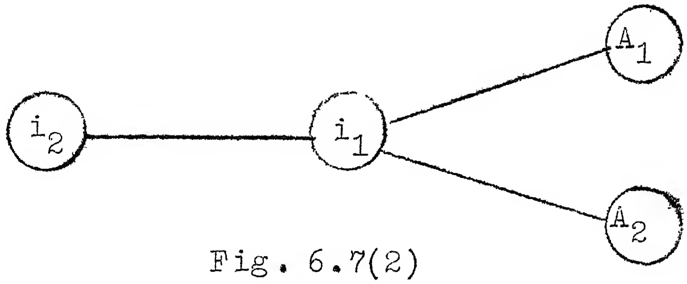


Fig. 6.7(2)

$A_1 \cup A_2 = \overline{\{i_1, i_2\}} = \{ j \mid j \in G$ and neither $j = i_1$ nor $j = i_2 \}$.

Total cost of communication over T_1 is

$$C(T_1) = r_{i_1 i_2} + r_{i_1 A_1} + r_{i_1 A_2} + r_{i_1 A_1} + r_{i_2 A_1} + r_{i_1 A_2} + r_{i_2 A_2} + Z_1 .$$

Total cost of communication over T_2 is

$$C(T_2) = r_{i_1 i_2} + r_{i_2 A_1} + r_{i_2 A_2} + r_{i_2 A_1} + r_{i_2 A_2} + r_{i_1 A_1} + r_{i_1 A_2} + Z_2$$

where for any node i of G

$$r_{iA_1} = \sum_{j \in A_1} r_{ij}$$

and $$r_{iA_2} = \sum_{j \in A_2} r_{ij}$$

where, $Z_1 [Z_2]$ is the sum of the cost of communication over all those edges of $T_1 [T_2]$ which are incident neither upon i_1 nor i_2. From the construction of T_2 it is easy to see that $Z_1 = Z_2$. Now there may be two cases:

(i) $$\sum_{j \in G} r_{i_1 j} = \sum_{j \in G} r_{i_2 j}$$

(ii) $$\sum_{j \in G} r_{i_1 j} > \sum_{j \in G} r_{i_2 j}$$

Case (i) Since $\sum\limits_{j\in G} r_{i_1 j} = r_{i_1 i_2} + r_{i_1 A_1} + r_{i_1 A_2}$

and $\sum\limits_{j\in G} r_{i_2 j} = r_{i_1 i_2} + r_{i_2 A_1} + r_{i_2 A_1}$

$$\therefore \quad C(T_1) = C(T_2)$$

Thus T_2 is an optimal communication spanning tree where i_1 is an inner node.

Case (ii) Since T_1 is an optimal communication spanning tree $C(T_1) \leq C(T_2)$. It implies

$$r_{i_1 i_2} + r_{i_1 A_1} + r_{i_1 A_2} + r_{i_2 A_1} + r_{i_2 A_2} + r_{i_1 A_1} + r_{i_1 A_2} + C_1$$

$$\leq r_{i_1 i_2} + r_{i_2 A_1} + r_{i_2 A_2} + r_{i_1 A_1} + r_{i_1 A_2} + r_{i_2 A_1} + r_{i_2 A_2} + C_2$$

$$\therefore \quad r_{i_1 i_2} + r_{i_1 A_1} + r_{i_1 A_2} \leq r_{i_2 i_1} + r_{i_2 A_1} + r_{i_2 A_2}$$

$$\therefore \quad \sum_{j\in G} r_{i_1 j} \leq \sum_{j\in G} r_{i_2 j} .$$

This contradicts the fact that $\sum\limits_{j\in G} r_{i_1 j} > \sum\limits_{j\in G} r_{i_2 j}$.

Hence node i_1 will be an inner node in at least one optimal communication spanning tree for $\{r_{ij}\}$.

Corollary (6.7.1) For any optimal communication spanning tree with an edge (i_o, j_o) such that node i_o is an outer node; following is true

$$\sum_{k\in G} r_{i_o k} \leq \sum_{k\in G} r_{j_o k} .$$

Proof. Follows immediately from the theorem.

The above result can be used for constructing an optimal communication spanning tree as follows:

If in the process of construction of a cut tree, we get an outer 'big-node' containing only two nodes say i and j, then there is no need of finding a minimum cut seperating i and j in $G(r_{ij})$. This is so because, now by comparing $\sum_{k \in G} r_{ik}$ and $\sum_{k \in G} r_{jk}$ we can decide which will be the outer node with the other as its neighbour in the final cut tree.

In Hu's algorithm [27], no preference is given for the order in which a pair of nodes should be selected for determining minimum cuts. It is quite possible that we may first get a minimum cut $(\{i_o\}, \{\bar{i}_o\})$ i.e., first we get an outer node i_o and then after solving one more maximum flow problem, we get its neighbouring node with the help of the minimum cut $(\{i_o, j_o\}, \{\overline{i_o, j_o}\})$.

Thus two maximum flow problems had to be solved to determine an outer node and the corresponding edge.

It would certainly be better if some how we first get $(\{i_o, j_o\}, \{\overline{i_o, j_o}\})$ instead of $(\{i_o\}, \{\bar{i}_o\})$ because with this cut, now by using the theorem (6.7.1), outer node can be determined without solving any more maximum flow problem.

Thus we can say that the number of maximum flow problems to be solved would quite often be reduced if 'would be' outer nodes are not branched out before their adjacent nodes. For

this we suggest the following selection procedure for branching from a big-node, or in the begining itself. To introduce edges between the nodes of a set N_o we select two nodes i_o, j_o from it as follows:

$$\sum_{j \in G} r_{i_o j} = \max_{i \in N_o} \left\{ \sum_{j \in G} r_{ij} \right\}$$

$$\sum_{j \in G} r_{j_o j} = \max_{i \in N_o} \left\{ \sum_{j \in G} r_{ij} \,\middle|\, \sum_{j \in G} r_{ij} \neq \sum_{j \in G} r_{i_o j} \right\}$$

Next we will show that in some cases, the theorem (6.7.1) together with the above selection procedure of nodes can reduce the number of maximum flow problems to be solved.

Suppose in the process of the construction of a cut tree we get a cut $(A_1, \bar{A}_1)$, where $A_1 = \{i_1, i_2, i_3\}$, and $R_{A_1} = R(A_1, \bar{A}_1)$

$$= \sum_{i \in \bar{A}_1} \sum_{j \in A_1} r_{ij}$$

and $R_{i_1} = R(\{i_1\}, \{\overline{i_1}\}) = \sum_{j \in G} r_{i_1 j}$

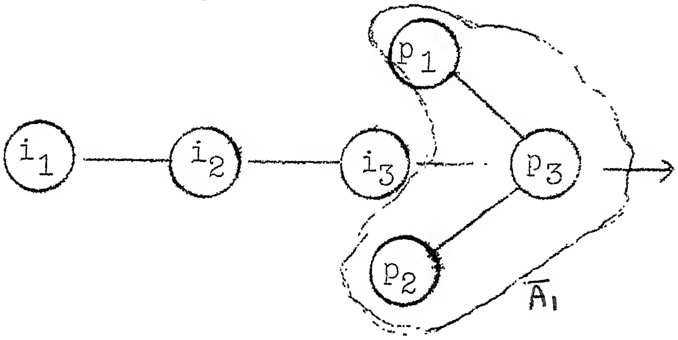
$R_{i_2} = \sum_{j \in G} r_{i_2 j}$

$R_{i_3} = \sum_{j \in G} r_{i_3 j}$

such that $R_{i_1} < R_{i_2} < R_{i_3}$.

With the help of the theorem (6.7.1) it can be shown that $R_{i_3} \geq R_{\bar{A}_1}$. This can be proved as follows: One node of A_1 say i_3 is going to be adjacent to the big node $\bar{A}_1$ (i.e., tree formed by the nodes of $\bar{A}_1$ will be incident upon i_3).

Now changing the positions of i_3 and $\bar{A}_1$ a better communication spanning tree can be obtained, which is a contradiction to the optimality of a cut tree.



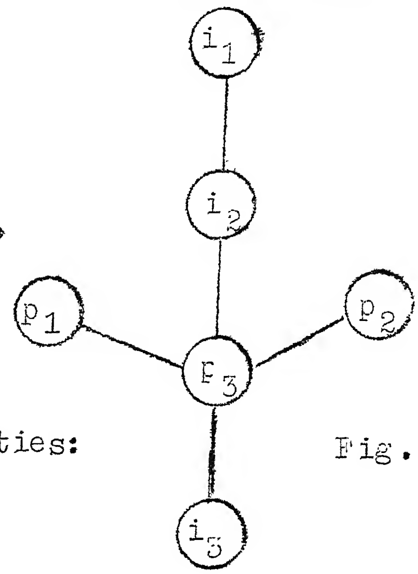


Fig. 6.7(3)

There are now only three possibilities:

(a) $\quad R_{\bar{A}_1} \leq R_{i_3}$ and $R_{\bar{A}_1} \geq R_{i_2}$

(b) $\quad R_{\bar{A}_1} \leq R_{i_2}$ and $R_{\bar{A}_1} \geq R_{i_1}$

(c) $\quad R_{\bar{A}_1} < R_{i_1}$

Case (a). Since i_3 is the only node of A_1 for which $R_{i_3} \geq R_{\bar{A}_1}$, therefore this will be the only node of A_1 which can be adjacent to the big node $\bar{A}_1$.

Again with the help of the theorem (6.7.1) we can say that the cut tree will take either of the following structures:

(i)

(ii)

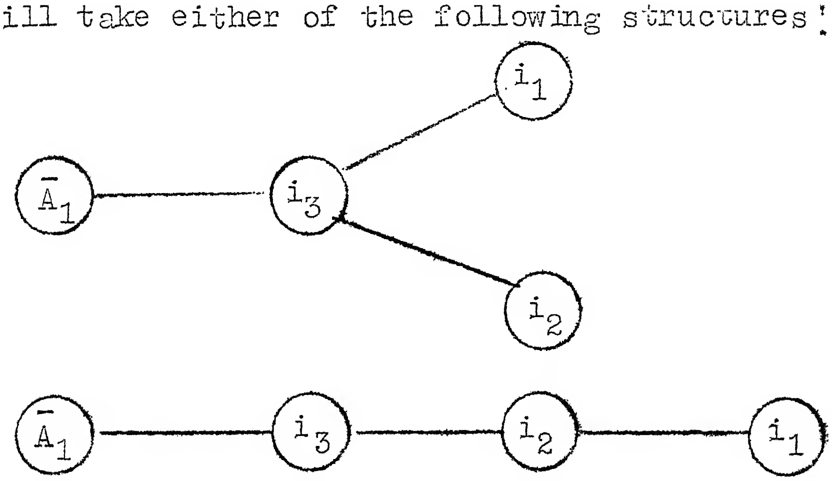


Fig. 6.7(4)

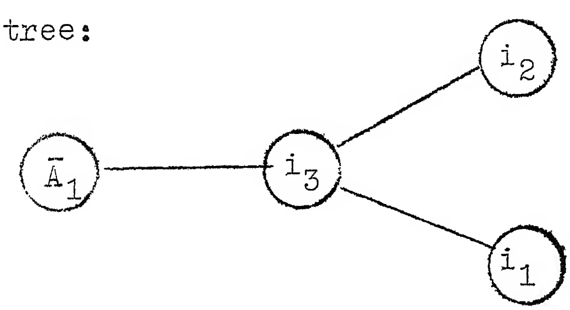
Since for the big node $A = \{i_1, i_2, i_3\}$, $R_{i_1} < R_{i_2} < R_{i_3}$, by the above suggested selection procedure, minimum cut seperating i_3 and i_2 will be determined first. As we know that this minimum cut is going to be present in the final cut tree if we arbitrary select nodes from $\{i_1, i_2, i_3\}$ for finding maximum flows, therefore by the above structures of the cut trees (i), (ii), we can say that the minimum cut seperating i_3 and i_2 in $G(r_{ij})$ will either be $(\{i_2\}, \{\bar{A}_1, i_1, i_3\})$ or $(\{i_1, i_2\}, \{\bar{A}_1, i_3\})$. From both the above cuts, big node A_1 can be opened without solving any other maximum flow problem. Thus in case (a) a big-node containing three nodes is opened up by solving one maximum flow problem instead of two.

<u>Case b</u> Since $R_{i_2} \geq R_{\bar{A}_1}$

and $R_{i_3} \geq R_{\bar{A}_1}$

Either of the nodes i_2 and i_3 can be adjacent to $\bar{A}_1$. Now again using the theorem (6.7.1) we can say that following are the only possible forms that the nodes of A_1 can take in the final cut tree:

(i)



(ii)

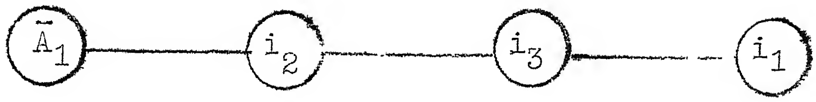
(iii) 

Fig. 6.7(5)

Therefore the minimum cut seperating nodes i_3 and i_2 (selected according to the above selection procedure) in $G(r_{ij})$ will either be

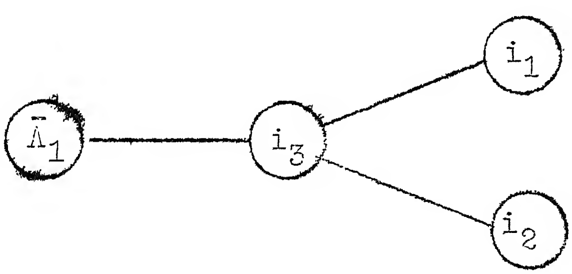
(i) $(\{\bar{A}_1, i_2, i_3\}, \{i_1\})$

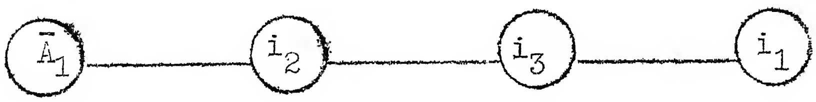
(ii) $(\{\bar{A}_1, i_2\}, \{i_3, i_1\})$

(iii) $(\{\bar{A}_1, i_3\}, \{i_1, i_2\})$

In all the above three cuts, big node A_1 can be opened up without solving any more maximum flow problem. Thus in this case also, big node A_1 is branched out by solving only one maximum flow problem instead of two.

Case (c) In this case, since $R_{i_1} \geq R_{\bar{A}_1}$, any node of A_1 can be adjacent to the big node $\bar{A}_1$. Using the theorem (6.7.1) we can say that the nodes of A_1 can take any of the following forms:

(i) 

(ii) 

(iii) 

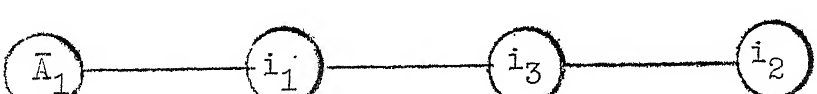
(iv) 

Fig. 6.7(6)

Therefore the minimum cut seperating nodes i_2 and i_3 will either be

(i) $(\{\bar{A}_1, i_2\}, \{i_1, i_3\})$

(ii) $(\{\bar{A}_1, i_3\}, \{i_1, i_2\})$

or (iii) $(\{i_2\}, \{\bar{A}_1, i_1, i_3\})$

In the cut sets (i) and (ii), node A_1 can be opened up without any extra computation. But for the cut set (iii), decomposition of A_1 can be either of the form (i) or (iv). Thus in this case only, we have to check, whether

$$R (\{i_1\}, \{\bar{i}_1\}) \gtrless R (\{i_2, i_3\}, \{\bar{A}_1, i_1\})$$

If in the begining, nodes of G are renumbered as $i_1, i_2 \ldots i_n$ such that $R_{i_1} \geq R_{i_2} \geq \ldots R_{i_n}$, the above selection procedure does not need any extra computational effort.

Hence in general, Hu's algorithm [26] can be modified by selecting nodes in the above specified manner.

Example 6.7

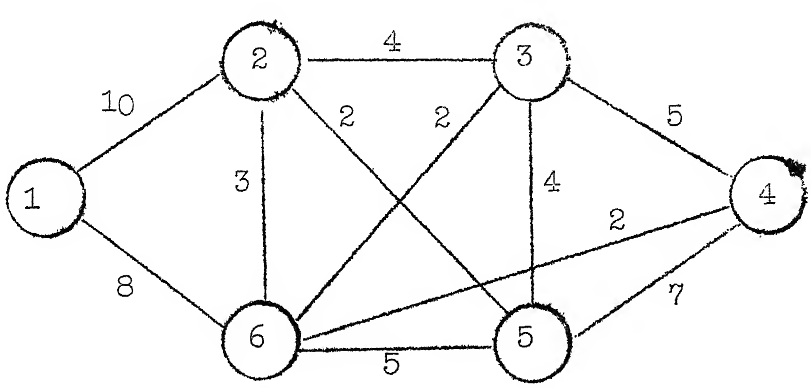


Fig. 6.7(7)

$$R_i = \sum_{j=1}^{6} r_{ij}$$

$R_1 = 18$, $R_2 = 19$, $R_3 = 15$, $R_4 = 14$, $R_5 = 18$, $R_6 = 20$.

Arranging the nodes such that their R_i's become in the non increasing order

6,2,1,5,3,4

We therefore first find maximum flow from node 6 to node 2 in $G(r_{ij})$. Minimum cut seperating 6 and 2 is ({1,2},{6,5,4,3}) having capacity 17. Therefore the starting structure of the cut tree is

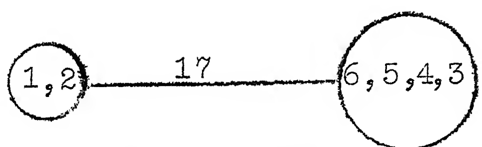


Fig. 6.7(8)

Since $R_2 > R_1$ node 2 can not be an outer node in the cut tree, therefore we can open the big node {1,2}

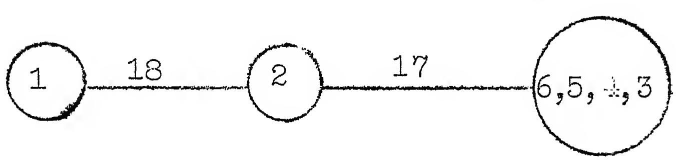


Fig. 6.7(9)

Now we solve a maximum flow problem selecting nodes 6 and 5. Minimum cut seperating these nodes in $G(r_{ij})$ is ({6,2,1}, {5,3,4}) which has the capacity 15.

Therefore form of the present cut tree is

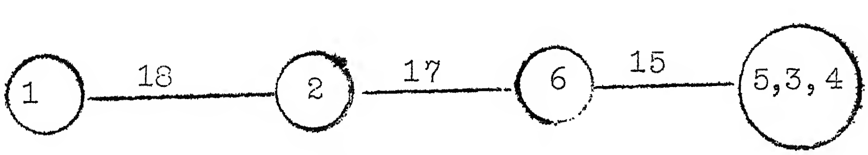
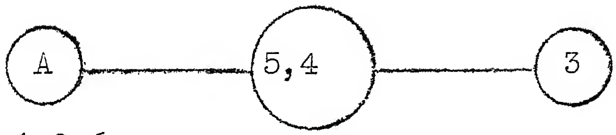


Fig. 6.7(10)

or



where A = 1,2,6 .

Since $R_4 < R_3$

$R_4 < R_A$

node 4 can neither be adjacent to A, nor it can be adjacent to node 3. Therefore the only possibility to open {5,4} is.

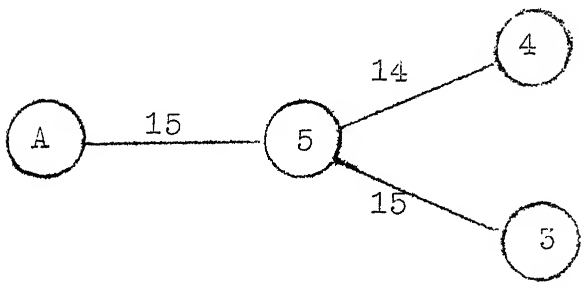


Fig. 6.7(11)

Hence the final cut tree is

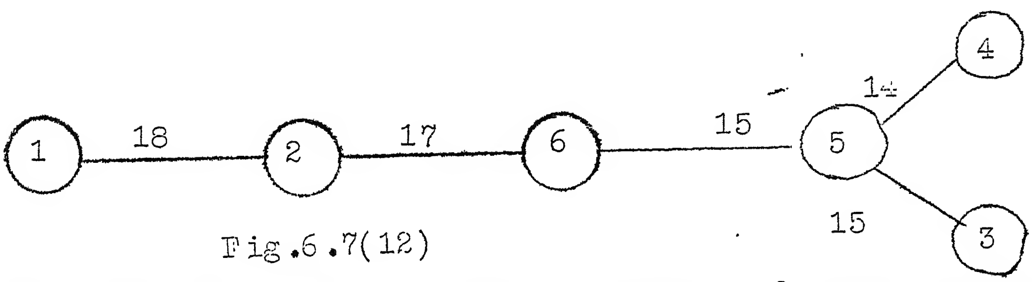


Fig.6.7(12)

Thus with the above selection procedure of nodes, we could get an optimal communication spanning tree (cut tree) only by solving three maximum flow problems, instead of 5.

REFERENCES

1. Adolphson D. and Hu T.C. : 'Optimal linear ordering' Siam J. Applied Maths. 25, pp 403-423 [1973].

2. Akopjan G.C. : 'Covering and matching of the vertices of a graph by the edges' U.S.S.R. Comp Math., Math. Physics 14, No.2, pp 184-192 [1975].

3. Alavi Y; Behzad, Lesniak M.,: 'Total matching and total Forster L.M., Nordhaus E.A. coverings of graphs' J. Graph Theory 1, No., pp 135-140 [1977].

4. Balas E. and Padberg M.W.: 'Set partioning: A survey' Siam Review 18,pp 710-760 [1976].

5. Balas E. and Samuelsson Haakon: 'A node covering algorithm' NRLQ, 24,2, pp213-233 [1977].

6. Balinski M.L. : 'Integer programming methods uses and computations' Management Science 12, pp253-313, [1965].

7. Balinski M.L. :'On maximum matching, minimum covering and their connections'' In H.W. Kuhn ed, Proceeding of the Princeton Symposium on Mathematical Programming Princeton Univ. Press, Princeton N.Y. pp303-312 [1970].

8. Berge C. : 'Two theorems in graph theory' Proceedings National Academy of Science U.S., 43, pp842-844 [1957].

9. Boesh F.T. : 'On covering the points of a graph with point disjoint paths'. Presented at the capital conference on graph theory and combinatorics Washington D.C. [1973]

10. Bondy J.A. : 'Extremal graphs with prescribed covering numbers' J. Comb Theory, Ser B, 24,pp 51-52 [1975]

11. Camerson, Peter J. : 'Minimal edge cover, colouring of complete graph' J. London Math. Soc. II, Ser. 11 pp. 337-346 [1975].

12. Chvatal V. : 'On certain polytopes associated with graphs' Centrede Recherches Mathematiques CRM 238, Universite de Montreal (Oct. 1972).

13. Dinic E.A. : 'Algorithm for solution of a problem of maximum flow in a network with power estimation'. Soviet Math. Dokl 11, pp1277-1280 [1970].

14. Berge Jack : 'Maximum matching and a polyhedron with 0-1 vertices' J. of Research of NBS B, Mathematics and Mathematical Physics 69B, pp126-130 [1965] .

15. Edmonds Jack : 'Paths, trees and flowers' Can J. Maths. 17, pp 449-467 [1965].

16. Edmonds J. and Johnson E.L. : 'Matching: A well wolved class of integer problems' Combinatorial structures and their applications, R.K. Guy, et. al. eds. Gordon and Breach, N.Y., pp89-92 [1970].

17. Even S. and Tarjan R.E. : 'Network flow and testing graph connectivity' Siam J. Comp. 4, pp507-518 [1975].

18. Ford L.R. and Fulkerson : 'A simple algorithm for finding maximal network flows and an application to Hitchock problem' Can. J. Maths 9, pp210-218 [1957].

19. Ford L.R. and Fulkerson : Flows in networks Princeton University Press Princeton N.J. [1962].

20. Fukurmaurat ABE.K. : 'An algorithm to solve minimum covering problem' Information Processing in Japan pp91-95 [1969].

21. Geoffrion A.M. : 'An improved implicit enumeration approach for integer programming' OR 17 pp 437-454 [1969].

22. Gomory R.E. and Hu T.C. : 'Multiterminal network flows'. Siam J. Appl. Maths. 9, pp551-571 [1961].

23. Goodman S., Hedetniemi: 'b-matching in trees' and Tarjan R.E. Siam J. Computing 5, no. 1 pp104-108 [1976].

24. Hern R., and Kuenzi H.P. : 'Computational methods for a special covering problem O.R. VI pp1-6 [1969].

25. Houck David, Vemunganti, and Rao R. : 'An algorithm for the vertex packing problem' O.R. Vol. 25, no. 5, pp773-787 [197-]

26. Hu, T.C. : 'Integer programming and network flow' Addison Vesley, Reading Mass [1969]

27. Hu, T.C. : 'Optimal communication spanning trees' Siam J. Computing 3, No. 3, pp189-19- [1974].

28. Ibaraki, Toshilide : 'Integer programming formulation of combinatorial problems' Discrete Mathematics, 16, pp39-52 [1976].

29. Jeurissa R.H. : 'Covers matching and odd cycles of a graph' Discrete Mathematics 13, pp251-260 [1975].

30. Karp R.M. : 'Reducibility among combinatorial problems' Plenu Press N.Y. pp85-103 [1972] by R.E. Mitter, J.W. Thachen and J.D. Dohlinger.

31. Karzanov A.V. : 'Determining the maximal flow in a network by the method of preflows' Soviet Math. Dokl 15, pp434-437 [197-]

32. Lawler E.L. : 'Covering problems, duality relations and a new method of solution'. Siam J. on Appl. Math. 14, pp1115-113- [1966].

33. Lawler E.L. : 'Combinatorial optimization networks and Matroids' Hold Rinchart and Winston.

34. Leuin, Mordechai : 'A note on Line Covering
Discrete Math. 5, pp283-285 [1973].

35. Leuin, Mordechai : 'Matching-perfect and cover-perfect graphs'
Israel J. Maths. 18, pp345-346 [1975].

36. Lorentzen L.C. : 'Notes on covering of arcs by nodes in an undirected graph'
OR-C-66-16 Univ. of California, Berkelay [1966].

37. Lovasz, L. : 'Covers, packings and some heuristic algorithms'
Proc. 5th Br. Comb. theory Aberdeen 1975, pp417-429 [1976].

38. Malhotra V.M. Kumar M.D. and Maheshwari, S.N. : 'An $O(|V|^3)$ algorithm for finding maximum flows in networks'
Information processing letters, 7, no. 6, pp277-278 [1978].

39. Meir A., Moon J.W. : 'Relation between packing and covering numbers of a tree'
Pacific J. Math. 61, pp225-233 [1975].

40. Neir A. : 'On total covering and matching of a graph'
J. of Comb. Theory Ser B, 24, pp164-168 [1978].

41. Mills W.H. : Covering problems
Proc. 4th Southeast Conf. Comb, Graph theory, computations, Boca Raton, pp23-52 [1973].

42. Nemhauser and Trotter : 'Properties of vertex packing and independence system polyhedron'
Math. Programming 6, pp48-61 [1974].

43. Nemhauser and Trotter : 'Vertex packing: structural properties and algorithms'
Math. Programming 8, pp232-248 [1975].

44. Niemieken Juhari : 'Some observations on covering of graphs'
Glasnik mat 111, Ser 10(30), pp3-8 [1975].

45. Norman R.Z. and Rakin M.O. : 'An algorithm for the minimum cover of a graph'.
Proceedings of the Amer. Math. Soc. 10, pp315-319 [1959].

46. Ore O. : 'Graphs and matching theorems' Duke Maths. J., 22, pp625-639 [1955].

47. Padberg M.L. : 'On the facial structure of set packing polyhedral' Mathematical Programming 5, pp199-215 [1973].

48. Perl J. : 'Graph algorithms and covering problems' Computing 13,2,pp107-113 [1974].

49. Picard J.C. and Queyranne : 'Vertex packing (VLP) reductions through alternate labeling' Tech Rept. EP 75-R-47 Ecote Polytechnique Montreal [1975].

50. Picard J.C., and Queyranne : 'On the integer valued variables in the linear vertex packing problem' Math. Programming 12, pp97-101 [1977].

51. Prim R.C. : 'Shortest connection networks and some generalization' Bell System Tech. J. 36, pp1389-1401 [1957].

52. Pulleyblank W.R. : 'Dual Integrality in b-matching problems' Discussion Paper 7717, Center for Operations Research and Econometrics, University of Louvain [1977].

53. Pulleyblank W.R. : 'Min node covers and 2-bicritical graphs'. Math Programming 17, pp91-103 [1979].

54. Steinson, Douglas : 'Determination of a packing number'. Ars Combinatoria 3, pp89-114 [1977].

55. Trotter L.E. : 'A class of facet producing graph for vertex packing polyhedra' Discrete Maths. 12,pp773-788 [1975].

56. White L.J. : 'A parametric study of matching and covering in weighted graphs' Tech. Rept. 06920-11-T System Engineering Laboratory. The Univ. of Michegen Ph.D. thesis [1967].

57. White L.J. : Minimum covering of fixed cardinality in weighted graphs. Siam J. of Appl. Math. 21, pp104-113 [1971].

58. White L.J., Gillenson Mark L. : 'An efficient algoritum for min k-covering in weighted graph'. Math. Progra mming 8, pp20-42 [1975].

59. Witzgall C. and Zahn C.Jr : 'Modifications of Edmonds' Maximum matching algorithm'. J. Res. NBS 69(B) pp130-135 [1965].

APPENDIX A

DLP [$\bar{G}(G_i)$;e] which is a maximum-flow problem on a bipartite graph, can be solved by any of the efficient algorithm of Karp [30], Fulkerson [18] and Malhotra [38]. Here we show that an optimal solution of its dual i.e. of LP [$\bar{G}(G_i)$;e] can be constructed directly from an optimal solution X of DLP [$\bar{G}(G_i)$;e], even if X is not an extreme point solution. Following is the procedure to generate an optimal solution of LP [$\bar{G}(G_i)$;e] from an optimal solution of DLP [$\bar{G}(G_i)$;e] :

Algorithm (A.1) : Let $N_1 \cup N_2$ denote the node set of the bipartite graph $\bar{G}(G_i)$.

Step 0 : Set $u_p^1 = 0$ if $\sum\limits_{q\in N_2} x_{pq} < c_p \ \forall\, p \in N_1$

Set $u_q^2 = 0$ if $\sum\limits_{p\in N_1} x_{pq} < c_q \ \forall\, q \in N_2$

Step 1 : (i) If $u_p^1 = 0$ and there is an edge $(p,q) \in \bar{G}(G_i)$ such that q has not been assigned a value, set $u_q^2 = 1$.

If $u_q^2 = 0$ and there is an edge $(p,q) \in \bar{G}(G_i)$ such that p has not been assigned a value, set $u_p^1 = 1$.

(ii) If $u_p^1 = 1$ and $x_{pq} > 0$, set $u_q^2 = 0$, provided node q has not been assigned any value.

If $u_q^2 = 1$ and $x_{pq} > 0$, set $u_p^1 = 0$ provided node p has not been assigned any value.

Repeat step 1, till no more nodes can be assigned values. If all the nodes have been assigned

values, stop; vector (U^1,U^2) with the components u_p^1, u_q^2 for $p \in N_1$ and $q \in N_2$ gives a (0,1) valued optimal solution for LP$[\bar{G}(G_i);e]$. Otherwise go to step 2.

Step 2 : Set $u_p^1 = 0$ for every node of N_1, which has not been assigned any value.

Set $u_q^2 = 1 \; \forall \; q \in N_2$, if there is an edge $(p,q) \in \bar{G}(G_i)$ such that $p \in N_1$ and has been assigned zero value, but $q \in N_2$ has not been.

Set $u_q^2 = 0$ for all the nodes of N_2 to which values have not been assigned till now.

Obviously all the components of (U^1,U^2) are non-negative. By the construction of the vector (U^1,U^2), there are only two possibilities; for (U^1,U^2) :

(i) It is infeasible for LP$[\bar{G}(G_i);e]$:

Infeasibility can occur only if some or more constraints are violated, i.e. there exists at least one edge (p_o,q_o) in $\bar{G}(G_i)$ such that $u_{p_o}^1 = 0$ and $u_{q_o}^2 = 0$.

(ii) It is not optimal for LP$[\bar{G}(G_i);e]$:

By construction of the vector (U^1,U^2) it is ensured that if the primal has a positive surplus

(i.e. $c_q - \sum_{p \in \bar{G}(G_i)} x_{pq} > 0$) for a constraint, then the corresponding dual variable is zero. Hence in case (U^1,U^2) is not optimal but is feasible for LP$[\bar{G}(G_i); e]$,

there will exist at least one constraint in the dual with the positive surplus such that the corresponding primal variable is also positive. Let (p_o, q_o) be this constraint such that $u^1_{p_o} = 1$, $u^2_{q_o} = 1$ and $x_{p_o q_o} > 0$.

We will now show that at any step of the algorithm (A.1) feasibility condition (i) and complementary slackness condition (ii) are not violated.

Step 0 : Since X represents a maximum flow on the bipartite graph $\bar{G}(G_i)$, we can not have any edge such that both the nodes on which it is incident, are unsaturated. Hence there is no edge (p_o, q_o) such that

$$\sum_{j \in \bar{G}(G_i)} x_{p_o j} < c_{p_o} \quad \text{and}$$

$$\sum_{j \in \bar{G}(G_i)} x_{j q_o} < c_{q_o}$$

i.e. $u^1_{p_o} = u^2_{q_o} = 0$ is not possible for any edge (p_o, q_o) of $\bar{G}(G_i)$.

Since in this step no node is assigned a value one, question of violating complementary slackness does not arise.

Step 1 : For the nodes p_o, q_o in case (i), following cases are possible :

(a) $\sum_{q \in \bar{G}(G_i)} x_{p_o q} < c_{p_o}$ and $\sum_{p \in \bar{G}(G_i)} x_{p q_o} < c_{q_o}$

(b) Either $\sum_{q \in \bar{G}(G_i)} x_{p_o q} < c_{p_o}$ or $\sum_{p \in \bar{G}(G_i)} x_{p q_o} < c_{q_o}$

(c) $\sum_{q \in \bar{G}(G_i)} x_{p_o q} = c_{p_o}$ and $\sum_{p \in \bar{G}(G_i)} x_{p q_o} = c_{q_o}$.

In case (a), flow X can be improved by sending

$$\min \{ c_{p_o} - \sum_{q \in \bar{G}(G_i)} x_{p_o q},\ c_{q_o} - \sum_{p \in \bar{G}(G_i)} x_{p q_o} \}$$ along the

edge (p_o, q_o) which contradicts the optimality of X.

In case (b) one of the nodes from p_o and q_o would have been assigned the zero value in step 0 and therefore the other node will be assigned value one in step 1. Hence again none of the constraints can be violated.

In case (c), since $\sum_{q \in \bar{G}(G_i)} x_{p_o q} = c_{p_o}$ and $\sum_{p \in \bar{G}(G_i)} x_{p q_o} = c_{q_o}$, neither p_o nor q_o will be assigned value zero in step 0. From step 1, since $u^2_{q_o} = 0$, there must exist an edge (p_1, q_o) in $\bar{G}(G_i)$ such that $x_{p_1 q_o} > 0$ and $u^1_{p_1} = 1$. Now $u^1_{p_1} = 1$ implies that there is an edge (p_1, q_1) such that $u^2_{q_1} = 0$.
In case $\sum_{k \in \bar{G}(G_i)} x_{kq_1} < c_{q_1}$ we stop.

If $\sum_{k \in \bar{G}(G_i)} x_{kq_1} = c_{q_1}$ we continue as before till we obtain a node q_s for which $\sum_{p \in \bar{G}(G_i)} x_{p q_s} < c_{q_s}$.

Due to the finiteness of the graph it is certain that we will be able to get such an unsaturated node q_s.

If $\sum_{q \in \bar{G}(G_i)} x_{p_1 q} = c_{p_1}$, there must exist an edge (p_1, q_1) having positive flow and $u^2_{q_1} = 1$. Now we proceed as before. Thus due to the finiteness of the graph $\bar{G}(G_i)$, we will be able to get a node p_s such that $\sum_{q \in \bar{G}(G_i)} x_{p_s q} < c_{p_s}$.

Similarly starting from node p_0, we can get a node q_t' such that $\sum_{p \in \bar{G}(G_i)} x_{p q_t'} < c_{q_t'}$. Thus a path of odd length from p_s to q_t' can be found such that both the nodes p_s and q_t' are unsaturated w.r.t. X. Thus as before, flow X can be improved by alternatively increasing and decreasing flows by a suitable $\epsilon > 0$, along the edges of this path. This will contradict the optimality of X.

Step 2 : By the description of the algorithm itself, we can say that both the nodes p_0 and q_0 of an edge $(p_0, q_0) \in \bar{G}(G_i)$ can not get zero values in step 2. Also for any edge $(p_0, q_0) \in \bar{G}(G_i)$ such that p_0 or q_0 gets value zero either in step 0 or 1, other node must also get a value one in step 1 only.

Again from the algorithm, it is easy to see that for any edge (p_0, q_0) such that $x_{p_0 q_0} > 0$, either both the nodes have been assigned values till the end of step one, or both remained unassigned.

Step 2 assigns values zero to all the nodes of N_1 which have not been assigned any value. Thus case (ii) will not occur in this step also.

Since the algorithm (A.1) is such that after assigning a value to a node in step 'i' it will not change its value in the following steps, therefore we can say that (U^1, U^2) thus obtained is feasible for LP $[\bar{G}(G_i); e]$ and also satisfies all the complementary slackness conditions with respect to X.

Hence (U^1, U^2) is an optimal solution for LP $[\bar{G}(G_i); e]$.

Date Slip

This book is to be returned on the date last stamped.
